A column generation approach for product targeting optimisation within the banking industry

J van Niekerk

orcid.org / 0000-0002-0761-959X

Thesis submitted in fulfilment of the requirements for the degree Doctor of Philosophy in Engineering with Industrial Engineering at the North-West University

Promoter: Prof SE Terblanche

Examination: February-March 2023

Student number: 23442212

Declaration

The entirety of the work contained in the electronically submitted thesis is my own original work. I affirm that I am the sole author of the information contained in this document and that the sources used in the thesis were referenced duly in the document's bibliography. The publication and reproduction of the document content by the North-West University of Potchefstroom will not result in the infringement of any third-party rights. The work has not been submitted for the obtainment of any other qualification whatsoever. It will only be submitted to fulfill the requirements for the degree: Doctor of Philosophy of Engineering in Industrial Engineering.

Date of Declaration: 22 February, 2023

Acknowledgments

Upon completion of the research study focused on solving the complex product targeting optimisation problem identified within the corporate financial setting, the following people and institutions should be acknowledged for their unwavering support and contributions towards the outcomes presented in this thesis:

- Prof SE Terblanche, at the North-West University of Potchefstroom, provided technical guidance on the theories relating to Dantzig-Wolfe decomposition and Column Generation and his support with the completion of this manuscript. Prof Terblanche also assisted in gaining access to the commercial solver Cplex, which allowed the author to develop the novel column generation framework, which was applied to solve the complex product targeting optimisation problem formulation discussed in this thesis.
- Annelie van Niekerk, my amazing and loving wife. Kristie van Niekerk, my beautiful daughter, and Jeanru van Niekerk, my newborn son, for their continued support and unconditional love during the three years of study. Annelie's encouragement and frequent words of wisdom empowered the author to push through the difficult times and overcome the daunting task of developing and contributing novel ideas to the research field of Industrial Engineering with a specific focus on Operations Research.
- My parents, Pierre and Christine van Niekerk, and Jannie and Linda Du Plessis, for their continued love and motivation. With such a support structure, the author successfully completed this research study. Their continued self-sacrifice toward their children and grandchildren is greatly admired and appreciated.

Most importantly, the author would like to thank God almighty for His continued blessings and the aptitude granted to the author to grow, develop and learn continually. Without His intervention and the insights, knowledge, and wisdom which He has bestowed upon the author, none of the work presented in this thesis would have been possible. *For the glory belongs to God alone!*

Abstract

Product targeting optimisation within the financial sector is becoming increasingly complex as optimisation models are exposed to an abundance of data-driven analytics and insights generated from a host of customer interactions, statistical and machine learning models, and new operational, business, and channel requirements. However, given the expeditious change in the data environment, it is evident that the product targeting formulation cited throughout the literature still needs to be updated to align with the realistic modeling dynamics required by financial institutions. In this thesis, an enhanced product targeting formulation is proposed that incorporates a large set of new modeling constraints and input parameters to try and maximise the economic profit generated by a financial institution. Furthermore, the proposed formulation ensures that the correct product is offered to the desired customers at the best time through their preferred communication medium. To solve the preceding product targeting formulation, a novel column generation approach can reduce problem complexity and, in turn, allow for significantly larger problems to be solved to global optimality within a reasonable time frame.

The column generation approach proposed in this thesis allowed the solution of complex product targeting formulations up to problem sizes consisting of 25000 customers, 35 products, and three channels. On the other hand, the standard branch-and-bound algorithm of Cplex could only solve problem instances up to a size of 5000 customers, 20 products, and 3 channels. The results show that the proposed column generation framework can significantly reduce the memory requirements of the various test instances, allowing it to solve significantly larger product targeting problems compared to standard solution methodologies. Furthermore, the solution framework reached close to global optimality for most of the test cases considered in this thesis.

Keywords

Product targeting; integer programming; Dantzig-Wolfe decomposition; column generation; optimisation; parallel processing; financial industry

Contents

1 Introduction **1**
- 1.1 Background and rationale . . . 1
  - 1.1.1 Marketing evolution . . . 2
  - 1.1.2 Response models . . . 2
  - 1.1.3 Customer profiling and clustering . . . 3
  - 1.1.4 Decisioning engines . . . 3
  - 1.1.5 Product targeting strategy . . . 3
- 1.2 Research scope . . . 3
  - 1.2.1 Research problem statement . . . 4
  - 1.2.2 Research purpose . . . 4
  - 1.2.3 Research objectives . . . 5
- 1.3 Research methodology . . . 5
  - 1.3.1 Literature review . . . 6
  - 1.3.2 Research method(s) . . . 6
- 1.4 Provisional chapter division . . . 7
- 1.5 Published article . . . 8

2 Problem description **9**
- 2.1 Direct marketing basics . . . 9
- 2.2 Data mining required for direct marketing . . . 10
  - 2.2.1 Data preparation and cleaning . . . 11
  - 2.2.2 Pattern tracking . . . 11
  - 2.2.3 Data classification . . . 12
  - 2.2.4 Association and correlation . . . 12
  - 2.2.5 Sequential patterns . . . 12
  - 2.2.6 Deviation detection . . . 12
  - 2.2.7 Clustering . . . 12
  - 2.2.8 Prediction . . . 13
  - 2.2.9 Prescriptive . . . 13
  - 2.2.10 Visualisation . . . 14
- 2.3 Product targeting optimisation data generation . . . 14
- 2.4 Campaign setup and execution . . . 15

3 Literature review **16**
- 3.1 Product targeting optimisation . . . 16
- 3.2 Related optimisation problems . . . 21
- 3.3 Concluding remarks . . . 23

4 Product targeting model formulation **24**
- 4.1 General notation . . . 24
- 4.2 Baseline IP formulation . . . 24
  - 4.2.1 Model objective . . . 25
  - 4.2.2 Model constraints . . . 25

4.3 Novel IP formulation . . . 27
4.3.1 Model objective . . . 27
4.3.2 Model constraints . . . 29
4.3.3 Novel IP formulation contributions . . . 35

5 Mathematical principles **37**
5.1 Dantzig-Wolfe decomposition theory . . . 37
5.2 Column generation theory . . . 40
5.3 Stabilisation techniques . . . 42
5.4 Dual smoothing . . . 44

6 Novel column generation approach **45**
6.1 Introduction . . . 45
6.2 Parallel processing . . . 45
6.3 Column generation solution framework . . . 46
6.3.1 Model starting heuristic . . . 47
6.3.2 Primal master problem . . . 53
6.3.2.1 Model objective . . . 53
6.3.2.2 Model constraints . . . 55
6.3.3 Dual master problem . . . 57
6.3.3.1 Model objective . . . 57
6.3.3.2 Model constraint derivation . . . 58
6.3.3.3 Reduced cost . . . 58
6.3.4 Sub-problem . . . 59
6.3.4.1 Model objective . . . 60
6.3.4.2 Model constraints . . . 60
6.3.5 Upper bound formulation . . . 61
6.3.6 Gap calculation . . . 62
6.3.6.1 Optimality gap . . . 62
6.3.6.2 Integrality gap . . . 62
6.3.7 Dealing with heading-in and yo-yo phenomenon . . . 62
6.3.8 Removing unused columns from algorithm . . . 64
6.3.9 Novel column generation model contributions . . . 64

7 Input data and verification testing **66**
7.1 Setting the scene . . . 67
7.2 Model data . . . 67
7.3 Generation of fictitious model data . . . 71
7.4 Problem scalability . . . 72
7.5 Novel IP formulation and column generation verification . . . 74
7.5.1 Novel IP formulation model verification results . . . 74
7.5.2 Novel column generation verification results . . . 76

8 Validation testing **80**
8.1 Novel IP formulation model validation results . . . 82
8.2 Novel column generation method validation results . . . 87

9 Model performance comparison **95**
9.1 Memory utilisation . . . 95
9.2 Optimality gap . . . 96
9.3 Row and columns added to models . . . 97
9.4 Column generation starting algorithm testing . . . 100
9.5 Iterative enhancements to novel column generation approach . . . 104
9.5.1 Reducing heading-in and yo-yo effects . . . 107

10 Summary and conclusion **110**
10.1 Chapter summaries 110
10.2 Future work 112

11 Appendix A: model parameters and variables **113**

12 Appendix B: baseline IP formulation summary **116**
12.1 Model objective 116
12.1.1 Model constraints 116

13 Appendix C: novel IP formulation summary **117**
13.0.1 Model objective 117
13.0.2 Model constraints 117

14 Appendix D: novel column generation approach summary **119**
14.1 Master problem 119
14.1.1 Model objective 119
14.1.2 Model constraints 119
14.2 Sub-problem 120
14.2.1 Model objective 120
14.2.2 Model constraints 120

15 Bibliography **121**

List of Figures

2.1 Data mining process flow for direct marketing campaigns (Jassim & Abdulwahid, 2021) 10

5.1 Block-angular Form, adapted from (Zhao, 2005) 38
5.2 Dantzig-Wolfe algorithm 40
5.3 Column generation algorithm 42
5.4 Column Generation Limitations (Vanderbeck, 2005) 43

6.1 Single Processing Example 45
6.2 Parallel Processing Example 46

7.1 Input data generation 72
7.2 IP formulation scaling analysis 72
7.3 Column generation scaling analysis 73
7.4 IP Formulation verification tests memory consumption 76
7.5 Column generation verification tests memory consumption 78

8.1 IP validation tests memory consumed (tight constrained) 84
8.2 IP formulation validation tests memory consumed (loose constrained), 2–6 85
8.3 IP formulation validation tests memory consumed (loose constrained), 8–12 86
8.4 Column generation validation tests memory consumed (tight constrained), 7–13 . . . 89
8.5 Column generation validation tests memory consumed (tight constrained), 15–17 . . 90
8.6 Column generation validation tests memory consumed (loose constrained), 2A–8B . 92
8.7 Column generation validation tests memory consumed (loose constrained), 10A–14 . 93

9.1 IP vs column generation memory consumption 95
9.2 IP vs column generation optimality gap 97
9.3 IP vs column generation column and row nalysis (tightly constrained) 99
9.4 IP vs column generation column and row analysis (loosely constrained) 100
9.5 Column generation starting algorithm memory consumption tests 103
9.6 Column generation iterative changes memory consumption tests 106
9.7 Column generation iterative improvements: test 10 108
9.8 Column generation iterative improvements: test 12 108

List of Tables

7.1 Fictitious input data example: customer, product and channel related parameters . . 67
7.2 Fictitious input data example: product related parameters 68
7.3 Fictitious input data example: channel related parameters 68
7.4 Fictitious input data example: product and channel related parameters 69
7.5 Fictitious input data example: customer, product, channel and options per channel . 69
7.6 Fictitious input data example: customer related parameters 70
7.7 Fictitious input data example: customer and channel related parameters 70
7.8 Fictitious input data example: customer, product and cross-sell related parameters . 71
7.9 Fictitious input data example: customer, product and cross-sell related parameters . 71
7.10 IP formulation and column generation model verification inputs (tight constrained) . 74
7.11 IP formulation model verification output (tight constrained) 75
7.12 IP formulation verification end state 76
7.13 Column generation verification output (tight constrained) 78
7.14 Column generation verification end state 79

8.1 IP and column generation solution validation input (tight constrained) 80
8.2 Column generation validation input 81
8.3 IP and column generation solution validation input (loose constrained) 82
8.4 IP solution validation output (tight constrained) 83
8.5 IP formulation solution validation output (loose constrained) 84
8.6 IP formulation solution validation end state 86
8.7 Column generation validation output (tight constrained) 88
8.8 Column generation validation output (loose constrained) 91
8.9 Column generation validation end state 94

9.1 Column generation starting algorithm input 101
9.2 Column generation starting algorithm output 102
9.3 Column generation starting algorithm end state 103
9.4 Column generation input 104
9.5 Column generation output 106
9.6 Column generation end state 107

11.1 Basic Model Parameters 113
11.2 Novel IP formulation model parameters 113
11.3 Model decision variables 114
11.4 Model indexes 114

List of Abbreviations

AMA: American Marketing Association

AMD: Advanced Micro Devices

CPU: Central Processing Unit

DCA: Dynamic Cell Abstraction

DMA: Direct Marketing Association

GAP: General Assignment Problem

GB: Gigabytes

GUI: Graphical User Interface

Ghz: Gigahertz

HRH: High-level Relay Hybrid

HTH: High-level Team Hybrid

IBM: International Business Machines

IDE: Integrated Development Environment

IP: Integer Programming

LHS: Left Hand Side

LP: Linear Programming

LRH: Low-level Relay Hybrid

LTH: Low-level Team Hybrid

MB: Megabytes

MIP: Mixed Integer Programming

ML: Machine Learning

MP: Master Problem

NP-Hard: Non-deterministic polynomial-time hardness

PI: Personal Information

PII: Personal Identifiable Information

POPIA: Protection of Personal Information Act

PSP: Pricing-sub Problem

PTP: Product Targeting Problem

RAM: Random Access Memory

RFM: Recency Frequency Monetary

RHS: Right Hand Side

RMP: Reduced Master Problem

ROI: Return On Investment

RPC: Right Party Contact

SKU: Stock Keeping Unit Problem

TODCM: Targeted Offers Direct Marketing Campaigns

UB: Upper Bound

USSD: Unstructured Supplementary Services Data

Chapter 1

Introduction

Product targeting optimisation within the banking industry is a problem that has gained significant traction in the last few years. This interest is due to institutions realising the immense financial benefit which could be unlocked by solving the desired problem. Even though it is a crucial problem to solve, limited research studies are available on this topic. The general approach to this type of NP-hard problem is to reformulate the product targeting problem into a simplified version of the main problem while leaving out critical business, channel, and product constraints or to apply heuristic algorithms to try and obtain a near-global solution. The preceding is why a novel column generation approach is proposed, which will solve a realistically sized product targeting problem to global optimality. To ensure the problem is representative of reality, novel business, product, and channel constraints are included in the problem formulation, increasing the complexity of the problem to be solved.

1.1 Background and rationale

Customers in the banking and retail industries are exposed to thousands of marketing communications daily since organisations compete heavily for customers' attention to grow their clientele base and outsmart the competition. These marketing communications provide information to customers about the services or products offered. The various marketing strategies available to these organisations must be used in complete agreement to deliver an effective and holistic message to the customer that will satisfy both customer and organisational needs. The marketing strategies aim to convince customers to acquire the services or products by ensuring that each customer is receptive to the messages and that the intent to purchase is established.

Modern-day marketing techniques are based on the premise that satisfied customers will provide positive feedback to their peers and will keep returning to use the organisation's offerings. This is why current marketing techniques ultimately aim to establish long-term customer relationships to drive customer growth. The American Marketing Association (AMA) reformulated the definition of marketing as a function within a system or organisation that manages customer relations through developing, communicating, and providing value to customers. This definition emphasises the importance of customer relationship management within the marketing environment and how imperative it is to have a well-constructed marketing strategy to meet set objectives (Du Plooy, 2012).

Marketing strategies need to impact the customer in some way to be effective, which means that the information presented to the customer must be relevant to their current interests, needs, and situation (i.e., personalised for each customer). Furthermore, these strategies' effect on customers should spur them to buy a product or procure a service. Therefore, a detailed discussion is provided throughout the subsequent sections to comprehend the complexity of constructing such an impactful marketing strategy.

1.1.1 Marketing evolution

There are generally two marketing strategies for product advertising and promotions that are used in practice: mass marketing and direct marketing. Mass marketing is a traditional marketing methodology that employs mass media to target current and potential customers with product-related information. The channels leveraged by this type of marketing strategy include radio, television, newspapers, and magazines. This strategy usually targets large groups of customers without discriminating between individual customers of a group. The information conveyed to these groups of customers is uniform, with each customer receiving the same message. This type of marketing strategy is quite expensive in comparison to direct marketing (Mitik, 2017).

Direct marketing differs from mass marketing because this strategy focuses on targeting individuals or households with personalised promotions. The Direct Marketing Association (DMA) defines direct marketing as a method for a company to use data systematically to achieve measurable marketing objectives by directly contacting current or potential customers to sell individualised offers. Direct marketing classifies or clusters customers in specialised segments to send promotional activities or personalised advertising to these specific classes of customers. Direct marketing strategies utilised channels such as email, SMS, and calls historically, with newer channels such as USSD, internet, and mobile push notifications introduced in recent years (Koopman, 2018). Direct marketing has been receiving increasingly more traction within various industries due to its high efficiency and effectiveness in guaranteeing improved success rates on marketing campaigns. When looking at the profit margin obtained from direct marketing campaigns across the years, it has been reported by Bose & Chen (2008) that it increased from $10 in 1999 to approximately $12.66 in 2005 for each $1 spent, with a steady increase ever since. In addition, the introduction of the internet and other digital channels has significantly lowered operational costs. This means that with even low response rates on campaigns, it is frequently still sufficient in ensuring the success of the various direct marketing campaign objectives.

Direct marketing campaigns typically consist of various activities, with the first interaction being solicitations sent to customers to sell either services or products. After receiving the solicitations, the next step is for the customers to decide whether to take up the product/service. After receiving feedback from the customers regarding their decisions (i.e., positive or negative), the marketers adjust their strategy to prepare for a new round of direct marketing activities (Bose & Chen, 2008). One of the critical processes to ensure the success of direct marketing campaigns is the correct selection of target customers to which these campaigns should be distributed. Target selection is generally preceded by sophisticated customer profiling and the development of response models generated by machine learning algorithms. The data produced by the target selection process must be optimised to maximise economic profit and satisfy various channel, business, and product constraints before contacting the various customers. Finally, the performance of the direct marketing campaign should be evaluated, whereafter, the results need to be fed back into the initial marketing strategy to ensure continuous improvement.

1.1.2 Response models

Response models such as statistical or machine learning models are utilised within the direct marketing domain to predict outcomes such as the probability of a customer taking up a particular product. Other aspects, such as determining the best communication channel or the preferred time of contact, can also be determined by these models. The mentioned techniques generate a score for each customer for the business problem under consideration (Lin, 2016). These model scores can take on binary values such as one or zero to indicate whether a customer will churn or not (Cohen, 2004). Another possibility is for the model score to return an integer value, indicating the number of products to sell to a customer. Lastly, these scores can also take on a continuous value between zero and one, representing a probability. Although the outcomes of these response models are typically utilised to answer a specific business problem, they cannot provide insight into different customer

clusters that may exist within the data under consideration. This is where customer profiling is introduced.

1.1.3 Customer profiling and clustering

Customer profiling uses customer pattern recognition and clustering to identify and make sense of customer segments (Liu *et al.*, 2012). Clustering algorithms are used to separate dissimilar customers from one another and group similar customers together (Bose & Chen, 2008). These techniques allow direct marketers to analyse small sub-segments of customers to identify corresponding relationships and characteristics which could be utilised to understand the dynamics between the various groups. Aspects that could be used to define clusters are customer age, demographic information, geographic location, product holding, the preferred channel of interaction, or historical purchase activity through different touchpoints (Camilleri, 2018). By interpreting those above, direct marketers can identify various customer personas within the existing customer base, with each persona having some preference for how they would like to be targeted. These customer personas are fed into the marketing strategy to guide the direct marketing decisioning process as to how and with what product these customers need to be targeted.

1.1.4 Decisioning engines

The data generated from the response and clustering models are structured as input data into either exact or heuristic optimisation algorithms. These algorithms are employed to construct marketing strategies through intelligent decisioning. In order to generate an optimal direct marketing strategy, several business, channel, and product constraints must be considered in addition to the response and clustering model outputs (Nobibon *et al.*, 2011). These include channel and resource limitations, channel and product exclusions, budget constraints, corporate hurdle requirements, cross-sell and upsell opportunities, campaign costs, and minimum quantity of products to sell per campaign, to name a few (Cohen, 2004). These aspects and many other constraints need to be factored into these types of optimisation models in order to generate a direct marketing strategy that can maximise a company's economic profit.

1.1.5 Product targeting strategy

An optimal targeting strategy should guide banking companies to:

1. propose the best product to sell to each customer,
2. at the right time of day,
3. through the customer's preferred contact channel,
4. while maximising profit and adhering to various business, channel, and product constraints.

The aspects mentioned throughout Sections 1.1.1 to 1.1.4 need to be considered. Therefore, for the remainder of this study, the focus will be on solving an optimisation problem known as the product targeting problem (PTP) with its application relevant to the direct marketing environment of a banking institution.

1.2 Research scope

Effective and personalised product targeting within the banking industry ensures revenue maximization. Personalised product targeting is, however, a complex process. A magnitude of aspects needs to be considered, before deciding on the best time of day, communication channel to use, and unique product the business wants to offer before reaching out to each customer. The complexity of this assignment problem is increased even further when considering a realistically sized customer base, such as seen in the banking sectors of South Africa. The intricacy and magnitude of this targeted

offering problem will make it practically impossible to utilise manual computations to obtain an efficient, targeted offering framework and simultaneously avoid sub-optimal solutions.

The objective of the product targeting problem is to increase the economic profit of the financial services provider. Profit maximisation can be achieved by generating revenue from an existing customer base or acquiring new customers. The former is "retention," whereas the latter is "acquisition." This study considers direct marketing campaigns where new products are offered to existing customers. Two aspects justify the focus of this study. Firstly, Reinartz *et al.* (2005) pointed out that firms should instead try and retain existing customers than attempt to acquire new ones. This is because prioritising expenditure towards retention will result in a more significant long-term impact on customer profitability. Secondly, the methods and models utilised within the financial industry for data analysis are better suited for retention efforts as banks have access to the deepest veins of customer data when considering their existing customer base. Although banks have access to a significant amount of customer data, Bernstel (2002) noted that many do not utilise these databases to their full potential when designing optimal product targeting models, which ultimately leads to sub-optimal results being obtained.

1.2.1 Research problem statement

Throughout the literature, it is evident that the product targeting problem is a highly complex problem defined as an NP-hard problem (Delanote *et al.*, 2013). Consequently, large-scale product targeting problems cannot be solved by only using exact methods for global optimality. Generally, researchers either reformulate the product targeting problem into a simplified version of the main problem leaving out critical business, channel and product constraints, or apply heuristic algorithms to try and solve this problem (Nobibon *et al.*, 2011). The concern with these approaches is that when reformulating the problem to a simplified version, the problem will not represent the real business problem and may result in sub-optimal results. When applying heuristic algorithms to solve the product targeting problem, these approaches generally result in random solution populations, which may only lead to generating local minima or maxima solutions. Generally, heuristic algorithms are also intertwined with the mathematical model it is trying to solve, which prevents these algorithms from being applied to variations of the same problem without requiring programmatic alterations.

1.2.2 Research purpose

This study aims to solve the direct marketing product targeting problem by exploiting an exact solution approach to compute a globally optimal solution. As part of the product targeting problem, the model would need to consider aspects such as campaign budgets, corporate hurdles, type of product to offer, channel, product, and resource limitations, including various other constraints discussed throughout the literature. This study aims to formulate and solve an optimisation problem representative of a realistic real-world product targeting business problem without leaving out constraints to simplify the complexity. The study is limited to direct marketing campaigns focusing on customer retention and does not consider aspects associated with customer acquisition.

The anticipated contributions that will be made throughout the thesis will include the following:

1. Formulate a comprehensive product targeting model considering various business, channel, and product rules that still need to be considered in the literature. Adding the preceding aspects to the model significantly increases the complexity of the optimisation model to be solved. The added rules are listed below:
    (a) Include aspects such as the best time of day to contact a customer when using a cell phone or landline as a contact medium.
    (b) Identifying the best possible number or email address from a list of sources for each customer to ensure that the highest possible right-party contact (RPC) rates are achieved per SMS, call, or email campaign.

(c) Including legislative constraints such as customer marketing consent and recency limitations used to override propensity scores.

(d) Selecting products to cross-sell given that the customer has taken up the primary offered product.

(e) Not only considering the best product to sell to the customer but also determining the best channel to utilise as a contact medium using propensity modeling results as input.

2. Expose the computational complexity of the newly proposed product targeting problem by evaluating the exponential increase in solving time when using exact solutions, as the problem is scaled to represent a realistically sized problem.

3. Design and implement an exact solution algorithm capable of solving the newly proposed product targeting problem to global optimality within a reasonable time. Utilising only heuristics to solve this type of problem would not allow the user to assess the integrality gap, which could lead to local optimums. Column generation algorithms at times also terminate at local optimums (potentially much better than heuristic solutions), but allows the user to gauge the quality of the solution.

4. Automate the model to allow for seamless interaction between the designed solution and the intended user. Automation will reduce implementation complexity and decrease time spent generating optimal direct marketing campaign leads.

1.2.3 Research objectives

The study objectives entail the following:

1. Provide a detailed i) review of the mathematical and computational advances made regarding the product targeting problem in the literature, and ii) discuss the mechanics associated with the proposed algorithms used in this thesis to solve the stipulated problem.

2. Construct a data handling tool to improve the process of fictitious data generation, acquisition, processing, and incorporation into the optimisation development environment using Python. Link the tool mentioned above to a graphical user interface (GUI) to increase ease of utilisation.

3. Formulate and design a product targeting problem in an interactive development software platform, capable of increasing the economic profit of a financial services provider while adhering to a multitude of business, product, and channel requirements.

4. Propose an exact solution algorithm capable of solving a realistically sized product targeting problem to global optimality within a reasonable time.

5. Verify the correctness and validate the effectiveness of the proposed product targeting problem in obtaining global optimal results for use in direct marketing campaigns within the banking industry. Also, determine the influence various modeling constraints, parameters, and data inputs might have on the problem complexity (i.e., solution time) and model results.

6. Identify and propose follow-up work associated with the work completed in this thesis that is worth being investigated when pursuing future studies related to the product targeting optimisation problem.

1.3 Research methodology

Throughout the research methodology, a high-level overview is provided of the essential literature topics that require investigation and the methods that will be needed to satisfy the research objectives. In addition, a systematic outline of the research methods is provided to ensure that each project objective, as stipulated in Section 1.2.3, is adequately addressed. Details about the preceding are provided in Sections 1.3.1 and 1.3.2.

1.3.1 Literature review

In this thesis, the literature review is presented in three parts and is covered in Chapters 2, 3, and 5. First, an introduction to the fundamentals of direct marketing campaigns is provided in Chapter 2, focusing on aspects such as direct marketing basics, data preparation, the theory behind machine learning/response models and the use thereof, customer profiling and clustering, and how campaign performance is measured. The preceding provides insights into the technical complexity around setting up a direct marketing campaign and the techniques used to derive data that will feed into the product targeting optimisation problem. Then, in Chapter 3, a detailed literature review is conducted on the solution methods applied by other researchers in trying to solve the product targeting problem. Emphasis is placed on the type of modeling constraints considered by these researchers and how they reformulated the problem to reduce model complexity. This review is used as a means of evaluating the advances made in the specific field of study, as well as to gain insight into the product targeting optimisation problem and how it is leveraged within the banking industry (Delanote *et al.*, 2011; Cohen, 2004).

Chapter 5 focuses on the algorithms applied to simplify the complexity of an NP-hard optimisation problem and assist in obtaining a global optimal solution. The preceding includes methods such as Dantzig-Wolfe decomposition, column generation solution algorithm, and stabilisation techniques. The purpose of the decomposition algorithm is to divide a highly complex optimisation problem into smaller sub-problems. This allows for solving multiple smaller sub-problems while obtaining the desired result for the complex overarching optimisation problem. Furthermore, this method reduces the computational capacity required to solve the problem (Ralphs & Galati, 2005; Desaulniers *et al.*, 2005).

1.3.2 Research method(s)

The following research methodology was applied in order to address the set-out project's objectives as outlined in this chapter:

1. Gain a fundamental understanding of the subject matter under consideration by conducting an extensive theoretical background and literature review related to optimisation solution algorithms and their applicability to the product targeting problem. By evaluating the preceding, an understanding can be obtained regarding the mathematical and computational advances that have been made in the related field of study. The study will also identify areas where improvements can be made to existing product targeting optimisation problems, as stipulated in the literature.

2. The data handling tool will be developed using Python 3.7.5 as a software package. This was used to generate and process fictitious data in such a manner as to ensure compatibility with the optimisation software used in this thesis. In addition, a GUI was designed and linked to the data handling tool to ensure improved and efficient data handling capability. Fictitious data generation is required to prevent the leakage of confidential banking data into the public domain.

3. The software package, Cplex (IBM Corp, 2015), in combination with C++ and Visual Studio 2019, was used as the interactive development platform to formulate the optimisation problem. A combination of heuristic and exact solution algorithms was leveraged to solve the product-targeting problem in a fashion. Aspects identified throughout the literature will be included in the product targeting problem formulation to ensure the comprehensiveness of the proposed model. Furthermore, many additional business, product, and channel rules will be added to the standard problem formulation to align the problem to the direct marketing dynamics within the banking industry.

4. The mathematical optimisation model was structured dynamically to allow the exclusion or inclusion of various business, product, and channel constraints to identify the effect each one

might have on the problem outcome. Moreover, an iterative approach was taken to introduce the additions made to the formulation and solution algorithm of the proposed product targeting problem.

5. The results from the iterative model were critically evaluated to determine the effect each addition might have on the computational solution time and overall model performance. Finally, a comprehensive mathematical model and exact solution algorithm were proposed, capable of reasonably solving a realistically sized product targeting problem to maximise economic profit.

1.4 Provisional chapter division

The introduction provides insight into the research purpose, objectives, and methodology to complete the study. This section is followed by nine chapters, a bibliography, and numerous appendixes.

Chapter 2 presents an introduction to the fundamentals of direct marketing campaigns, with an overview of the product targeting problem and its applicability to the banking industry. In addition, technical aspects related to the formulation, setup, and execution of direct marketing campaigns are provided to gain a vital understanding of the dynamics associated with the problem to be solved.

As seen throughout the literature, optimisation techniques applied to the product targeting problem are presented in Chapter 3 to acquire knowledge of the solution methodologies applied in solving the problem. Specific focus is set on exact solution algorithms and how they can be utilised to try and reach global optimality.

The mathematical formulation of the product targeting problem is presented in Chapter 4 using an iterative approach, with basic terminology and notations being defined to ensure the simplicity of the model derivation. Descriptive reasoning as to why the model objective and constraints were added to the model formulation is provided with a specific focus on the included business, product, and channel constraints. Detailed examples regarding the mechanics between various constraints, as well as the effect each might have on the complexity and computability of the model in reaching a global optimal solution, are discussed. The non-approximability of the proposed product targeting problem when applying exact solution algorithms will also be presented.

A discussion is provided in Chapter 5 regarding the optimisation theory applied throughout the thesis to solve the product targeting problem. The preceding includes algorithms such as the Dantzig-Wolfe decomposition and column generation algorithms. In addition, simplistic examples are included to fundamentally understand the theory and computational dynamics behind the applied methods.

Chapter 6 introduces the mathematical formulation of the exact solution applied to the product targeting problem to allow the problem to be solved. The algorithm is introduced iteratively, with the emphasis being placed on the complexities associated with combining heuristic and exact formulations in order to obtain global optimality. The improvement of the model solution time and memory consumption when implementing the algorithmic framework is discussed in detail.

Computational results obtained from the product targeting problem are presented in Chapters 7 and 8 as a way of verifying and validating the efficacy of the newly proposed exact solution model in solving different variations of the product targeting problem. The results from the exact solution approach are compared to traditional exact solution approaches to determine the potential improvement. The capability of the proposed model in solving realistically sized instances of the product targeting problem is also investigated. The input data required to solve the product targeting problem are elucidated, and a high-level overview of the data acquisition and processing tool is supplied.

Chapter 9 provides some comparative insights as to the performance of the IP formulation when compared to the column generation algorithm and how the solution capability of these algorithms

changes when exposed to varying sizes of product targeting optimisation instances. A view is also provided on how the algorithmic improvements proposed in this thesis have enhanced the performance of the column generation algorithm.

The thesis concludes in Chapter 10 with a summary of the work conducted. A synopsis of the contributions made to the field of study is given, and significant findings identified are discussed. Future work related to the product targeting optimisation problem is also proposed.

The thesis is supplemented with accompanying appendixes containing summarised mathematical formulations of the exact, decomposed formulations and the standard formulation obtained from the literature for comparative purposes. The appendixes will provide a summarised view of all technical aspects covered in this study to allow the reader a transparent view of the contributions made.

1.5 Published article

In fulfilment of the requirements for obtaining a PhD within the Faculty of Engineering at the North-West University, the author compiled and published (12/22/2022) an article related to the work contributed in this thesis in a South African journal known as ORiON. For more detail regarding the article's content, refer to (Van Niekerk & Terblanche, 2022).

Chapter 2

Problem description

Chapter 2 unpacks the importance of direct marketing within a retail banking ecosystem to understand the importance of solving the product targeting problem addressed in this thesis. The preceding will also provide the reader with some context regarding the complexities associated with collecting, cleaning, and generating the data required as input into the product targeting optimisation problem.

2.1 Direct marketing basics

Direct marketing forms part of a critical capability required within a retail banking setting to retain existing customer relationships while aspiring to acquire new ones (Cetin & Alabas-Uslu, 2017). As a result of the product offerings available within the retail banking industry, there needs to be more room for differentiation between competitor offers. Therefore the critical driver for success or failure dramatically depends on the effectiveness of a retail bank's direct marketing strategy. As a result of customers being bombarded with various marketing initiatives from retail brands, they become less susceptible to these marketing attempts and learn to ignore them completely. In recent years, the retail industry shifted its focus towards personalised direct marketing to counteract this phenomenon. Previously, marketing was seller-focused, where marketing practitioners were more concerned with selling as many products or services to whoever could be reached. The message of the seller-focused approach was leaning towards what the company could offer or what products they could sell. The personalised direct marketing approach has adopted a consumer-focused or, also known as a customer-centric approach (Bhaskar *et al.*, 2009). This approach entails using customer data to customize product offerings on an individual customer level. The consumer-focused approach tries to determine customers' needs, wants, lifestyles, social classes, or purchasing behaviors to propose the best-suited product or service to every customer. Personalising the proposed offers significantly increases campaign efficiency and consequently reduces cost. This personalisation is made possible due to the capability of retail banks to gather tremendous amounts of data from client activities. The data generally originates from either customer-provided information such as name, surname, age, gender, and many more, or customer activity data such as transactional or user interaction/response data. As mentioned in Chapter 1, direct marketing is generally executed using SMS, call, e-mail, or newer channels such as USSD, internet banners, in-app messages, or mobile push notifications.

An exhaustive list has been provided by Mai (1997), allowing the reader to comprehend the core distinctions of direct marketing to that of mass marketing:

1. Direct marketing allows marketers to identify, select and target customers most likely to respond positively to the marketing interaction. By selectively targeting a selected customer or audience, the number of marketing attempts is significantly reduced, improving cost-effectiveness.

2. Given that a selected grouping of customers is targeted, it allows marketers to highly person-

alise, and structure offers according to the preferences of these customers. The preceding leads to higher response rates and assists the retail bank in achieving the set-out objectives.

3. The robust marketing strategies formulated by marketers allow the ability to bridge linguistic and cultural barriers while conveying detailed knowledge and information to interested customers.

4. Direct marketing allows marketers to also effectively measure and track the performance of sales promotions and identify the best strategy to employ for future interactions with individual customers.

5. Direct Marketing also enables long-term relationship management, customer trust, and the cultivation of brand loyalty.

As alluded to in the above paragraph, the fundamental pillar required by retail banks to formulate a robust marketing strategy is a marketing database containing a host of customer information, interactions, and responses (Cohen, 2004). To effectively leverage the preceding information to derive actionable insights that could feed into a product targeting optimisation problem formulation, a process known as data mining is required. Data mining is discussed in the next section.

2.2 Data mining required for direct marketing

A diagrammatic representation of the data mining process is portrayed in Figure 2.1. The preceding is a typical process flow followed by a financial institution to reach a mature enough point where the implementation and deployment of prescriptive models, such as the product targeting optimisation problem, are viable within a direct marketing campaign environment. Note that the portrayed process flow seen in Figure 2.1 is not a catch-all end-all to the data mining domain, as various other techniques and methodologies can be added to enhance the explanation of the process.

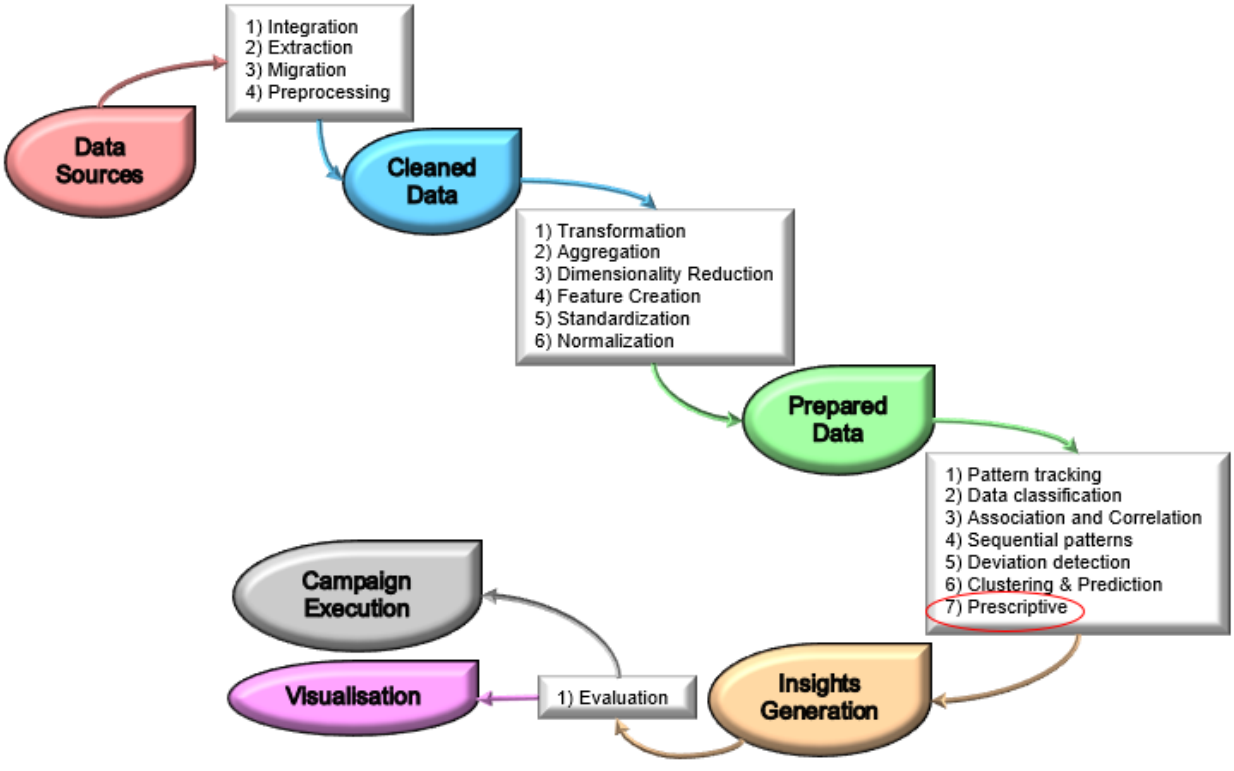

Figure 2.1: Data mining process flow for direct marketing campaigns (Jassim & Abdulwahid, 2021)

The following sections provide introductory information regarding the various aspects of the data mining process flow to ensure the reader has a sound basis for understanding the complexities of

setting up such an advanced product targeting problem formulation as proposed in this thesis. In addition, insights are provided into the various data mining techniques an institution would need to leverage to generate data for use within a product targeting model. Finally, in Figure 2.1, the prescriptive analytical process is encircled to emphasise that the work conducted in this thesis proposes contributions to this research and modeling domain.

Data mining is a computer-driven process of cleaning, transforming, and extracting valuable knowledge and insights from data while using the previous to guide and influence strategic strategies and decision-making (Jain & Srivastava, 2013). This process allows companies to access vast volumes of data to obtain valuable information at the right time. Modern data mining techniques entail using various statistical, mathematical, machine learning, and artificial intelligence algorithms to generate insights from vague, noisy, and incomplete data sets and identify valuable and predictable patterns essential for decision-making. Formulating data-driven strategies is crucial in allowing retail banking companies to succeed in an increasingly competitive market. The type of data mining techniques used is generally industry-specific, with some problems requiring the application of only data cleaning and preparation techniques for visualization purposes. In contrast, others require more cutting-edge artificial intelligence and exact optimisation algorithms to derive the required insights.

2.2.1 Data preparation and cleaning

The initial step in any data mining process is to clean and prepare the data. This process is vital as data is required to be formatted and cleaned before it can be utilised in different analytical methods. Cleaning the data entails various aspects, such as data integration, extraction, migration, and preprocessing. Once data cleaning has been completed, data preparation techniques such as transformation, aggregation, modeling (i.e., dimensionality reduction and feature creation), or standardization and normalization techniques could be applied in preparation to derive insights from the data (Jassim & Abdulwahid, 2021). The preceding steps are essential in understanding the attributes and features contained within the data. The previous will also assist end users in identifying the best use for the data under consideration. The data is rendered unreliable and meaningless without applying the first step of data mining. Following the proper data cleaning and preparation procedures allows an organisation to also adhere to proper data governance standards. Once data has been cleaned and prepared correctly, the next step is to generate insights from the underlying data. Various methodologies can be utilised to achieve this outcome. For example, one could use simplistic methods such as pattern tracking or association/correlation identification or more advanced techniques such as predictive and prescriptive analytics. Throughout the subsequent sections, insights are provided regarding each of the methodologies listed in Figure 2.1, with a specific focus on prescriptive analytics, as this forms the baseline for this thesis.

2.2.2 Pattern tracking

Pattern tracking is the process of monitoring trends and identifying patterns within data that could be used to derive actionable insights (Rygielski *et al.*, 2002). Once a pattern is identified within a dataset, it can be leveraged by an organization to guide its thinking around the strategic goals and outcomes (Jassim & Abdulwahid, 2021). For example, when focusing on a retail company, a possible pattern that could be identified is that the sales of a particular product are significantly more than other related products. The retail company can leverage this insight to either augment its marketing strategy for the remaining products or potentially design new products following a similar framework as the top-selling product. Pattern tracking can also be leveraged to determine a noteworthy change in customer or data distributions, which could hint toward some action that needs to be taken. The insights generated employing pattern tracking can be utilised to generate predictive features when applying machine learning and artificial intelligence algorithms, as discussed in Section 2.2.8.

2.2.3 Data classification

Data classification involves analysing several datasets within an organization in order to identify various attributes linked to each one of the datasets. Once the main attributes of a dataset(s) are identified, organisations can utilise this information to classify and categorise the type of data stored within their various databases. Classifying data is essential as it allows an organisation to identify various types of information, such as sensitive personal identifiable information (PII), personal information (PI), and digital or technology information. By correctly classifying information, organisations can use the information for what it was intended for or redact, for example, personal information from documentation, to protect the exposure of client information to the public domain. Data classification techniques also allow organisations to identify which datasets can be utilised when constructing clustering, prediction, and prescriptive models (Jassim & Abdulwahid, 2021).

2.2.4 Association and correlation

Association is a statistical data mining technique used to identify data or events linked to one another. In other words, the association process is to find item sets that frequently occur together. It is similar to the principle of co-occurrence in machine learning, where a particular data-driven event indicates another activity that will take place. The principle of correlation is also similar to that of association, with correlation referring to a relationship between two data points. Identifying association and correlation rules within a dataset enables an organisation to predict occurrences or events intelligently. These data attributes are part of the baseline process used for feature generation in machine learning and clustering models (Raval, 2012). In the literature, we note that some of the methods used for association mining include the Apriori, FP-growth, Eclat, and AprioriDP algorithms (Lin, 2016; Ngai *et al.*, 2009).

2.2.5 Sequential patterns

Sequential pattern mining is identifying a series of events that takes place in a predefined sequence. Sequential pattern recognition is an extension of the association rule theory. This methodology is generally applied within the transactional data mining domain where, for example, an organisation can identify which items a customer will buy after an initial transaction. Furthermore, by leveraging the preceding methodology, organisations can recommend additional products that could increase sales (Raval, 2012). Algorithms generally applied to achieve the previous include the AprioriSome, DynamicSome, AprioriAll, FreeSpan, MAPres, PrefixSpan, and GSP, to name a few (Lin, 2016).

2.2.6 Deviation detection

The process of deviation or outlier detection entails the identification of aberrations within a dataset that could be identified as an anomaly. An outlying point could create undesired biases when applying classification or clustering algorithms, resulting in inaccurate predictions or proposed clusters. Outliers could also provide organisations with interesting insights that could point to a change required within a business or operational process. Such an example would indicate a fraud detection incident within the banking industry. Accurately identifying outliers within an organization's dataset allows the organisation to prepare for future occurrences to satisfy set-out business objectives. In the literature, control charts, linear regression, and Manhattan distance techniques were applied to identify deviations or outliers within data (Bakar *et al.*, 2006; Agrawal & Agrawal, 2015).

2.2.7 Clustering

Clustering is an analytical technique concerned with dividing data points into several groups, which are seen to be more similar to data points in the same group and dissimilar from other groups. In simple terms, it is the process of segregating data into groups containing similar traits and allocating the preceding into a cluster. Clustering leverages visual techniques to understand data distribution in relation to other data points. Organisations can utilise these visual depictions to identify trends

that could lead to actionable business insights. Clustering or segmentation techniques such as k-means, DBSCAN, BIRCH, Mean Shift, and a few others can be leveraged to group similar customers into a cluster (Berkhin, 2002). The clustering technique was widely researched and implemented by researchers to reduce the computational complexity of the product targeting problem, as noted in Chapter 3. Instead of optimising the allocation of products and channels for millions of customers, researchers utilised clustering techniques to aggregate customers into, for example, 10000 clusters. In doing this, the number of decision variables to be determined is significantly reduced as well as the computational complexity of the problem is simplified (Raval, 2012; Ngai *et al.*, 2009).

2.2.8 Prediction

The prediction methodology is an effective technique within data mining that is used to evaluate patterns in historical and current data to extend it to potential future events (Rygielski *et al.*, 2002). Predictions are typically utilised within organisations when they are concerned with what will happen in the future. Something typical is the probability of a customer answering his/her phone or the likelihood of a customer purchasing/taking up a product. Prediction methodologies can be classified as either machine learning techniques such as decision trees, logistic regression, random forests, and gradient boosting models or artificial intelligence methods such as neural networks, natural language processing, machine vision, and voice recognition (Jain & Srivastava, 2013). The preceding methodologies can be implemented as either regression or classification algorithms. Within the product targeting context as proposed in this thesis, prediction algorithms would be required to generate various probability measures that will pose as input data into the optimisation model (Ngai *et al.*, 2009). More detail regarding the probabilities scores leveraged in the product targeting formulation is discussed throughout Section 2.3 and Chapters 4, 6, and 7.

2.2.9 Prescriptive

The next phase of the data mining process is the implementation of prescriptive analytical methods to guide the decisions performed by an organisation (Bengio *et al.*, 2018). Contrary to predictive analytical methods, which are concerned with identifying what might happen in the future, prescriptive methods are designed to present end users with the best options to follow to reach a set business objective. Implementing these methods starts with constructing a mathematical model representative of the business or organisational environment. The model takes as input both historical and predictive analytical data to interrogate the best possible decisions to perform while adhering to various modeling criteria. The end goal of these modeling techniques is either to maximise or minimise a business objective function such as profitability or customer lifetime value, for example. This analytical methodology is superior to the previously mentioned modeling techniques in that it does not just provide users with some insights into what might happen (i.e., predictive models) but provides actual tangible guidance as to what decisions need to be taken to achieve optimal outcomes. Algorithms applied to solve these prescriptive models are classified under two main categories known as exact and heuristic solution approaches. Exact algorithms allude to a combination of mathematical methods used to model a specific business problem and the use of exact algorithms such as the simplex or branch-and-bound methods to solve said problems. Heuristic algorithms are rule-based modeling techniques used to find good approximated answers to business problems. Exact solution algorithms are focused on finding the global optimal solution to a given problem, whereas heuristics are often only able to find local optimums. Implementing the preceding techniques is highly dependent on the size and complexity of the problem to be solved. The product targeting problem that is the focus of this thesis falls within the prescriptive domain. Some of the data mining techniques, as discussed in Sections 2.2.1 – 2.2.8, are leveraged to formulate this problem. The techniques used in this thesis to prepare the input data for the product targeting optimisation problem, as seen in Chapter 7, will be elaborated on in Section 2.3.

2.2.10 Visualisation

The final methodology leveraged within the data mining process is known as data visualisation. This process entails displaying insights and knowledge obtained from applying various statistical, mathematical, clustering, predictive, or prescriptive techniques. Visualisation is generally portrayed in charts, graphs, images, tables, matrices, or trees to allow end users to quickly understand and make sense of complex processes or analytical outcomes. The previous is also a way to quickly quantify the monetary gain achieved when deploying, for example, a prescriptive product targeting model within a financial institution's ecosystem, as well as convey the relevant information to executive management (Ngai *et al.*, 2009).

2.3 Product targeting optimisation data generation

As will be discussed in Chapter 4, we propose a product targeting formulation that makes use of various probabilistic input data such as the following:

1. the probability of getting hold of a customer via a specific communication channel,
2. the probability of talking to the right party contact once a customer was successfully reached,
3. the probability of reaching a customer telephonically for various periods of the day.

For this study, the preceding data points were already generated and available to the end user as input into the product targeting optimisation model. However, in reality, an extensive data mining process similar to what was proposed in Figure 2.1 would have been followed to generate the required insights, with a specific focus on the predictive modeling theory, as seen in Section 2.2.8. The first step would have been to identify the various data sources required to construct predictive models capable of answering the above business questions. Data such as customer contact details, historical call logs and their associated dispositions, customer profiles, demographics, product holdings, and so forth would be required to set up said models. After identifying the data sources, the developer would be required to identify in which databases the relevant sources are stored and how they could be incorporated into an integrated development environment (IDE), enabling end users to clean and prepare the data. Data cleaning would entail considering aspects such as integration, extraction, migration, and preprocessing.

Once the data has been cleaned, the developer would need to use data preparation techniques such as feature creation, normalization, and transformation to create relevant training and test data sets that would need to be fed into the relevant ML models, allowing the respective models to be trained. Upon completion of ML training, the developer would need to evaluate the ML model performance using metrics such as mean absolute error or root mean squared error in the case of regression models or receiver operating, lift, or cumulative gain curves when using classification models. Identifying the best-suited ML model to solve the relevant business problems is generally an iterative process. Various models such as decision trees, random forests, gradient boosting, neural networks, and many others are applied to solve the stated problem. During the model evaluation, the performance metrics for the various models, as stated above, are compared to identify the optimal model framework to utilise. Methodologies such as cross-correlation or hyperparameter tuning could also be applied to improve ML model performance. Once the ML model training process has been completed, the various models can be utilised to predict the desired probabilistic outcomes. The generated ML model results are then utilised as input data into the product targeting problem to guide the prescriptive model to perform optimal decisions. The probabilistic predictions are dynamic and might change as customer profiles, and features are augmented over time. It is, therefore, essential to deploy and automate the desired ML models to refresh the predicted values and serve the outcomes of the product targeting optimisation model on a daily or monthly basis. With dynamic changes in the input data, the product targeting optimisation model would need to be executed frequently to ensure that sub-optimal decisions are avoided.

The other input parameters utilised within the product targeting optimisation formulation are generated from customer interactions, product specifications, and compliance guidelines or business, operational, and channel requirements stipulated by the financial institution. These data points are significantly easier to obtain or generate compared to the previously mentioned probabilistic model insights. From a data mining perspective, it only entails using the data cleaning and preparation steps depicted in Figure 2.1, with no significant processing or transformations required to ingest these data points into the required prescriptive models. After generating all of the data sets essential for developing a product targeting optimisation model, the next step in the process would be to develop, deploy, and act on the insights ascertained from the prescriptive model. From a financial institution perspective, lead actioning would take the form of an official campaign charter process. The last step in the direct marketing framework is to present customers with personalised offers generated from the product targeting optimisation model.

2.4 Campaign setup and execution

The insights generated from a product targeting optimisation model will guide campaign managers in acting on the respective leads. For example, from the product targeting model output, it would be possible to identify which product should be offered to what customer. The relevant prescriptive model will also propose additional detail regarding the time of day and the use of a specific channel. Developing such a product targeting optimisation model allows financial institutions to automate and scale their campaign environment, which means that without the implementation of such an advanced decision engine, the onus will lie on the company's employees to manually scour through the various data sources and insights to develop a campaign strategy. The preceding will require manual computations where the institution will be limited on the number of campaigns designed to interact with customers. Such a manual process will also result in sub-optimal decisions, resulting in a steady decline in customer sentiment.

Even though these prescriptive product targeting models exist, financial institutions find themselves in a situation where campaigns consist of millions of customers, with each customer expecting to receive individualised and personalised offers designed explicitly for their wants and needs. As noted in Chapter 3, the existing prescriptive models are designed to account for only some of the business, operational, and channel requirements that modern financial institutions face. Secondly, exact optimisation methodologies capable of solving the product targeting problem to global optimality do not exist when considering a customer base exceeding a million customers and many products. The current exact optimistation techniques either run into memory limitations or are significantly constrained on computational time. The alternative is to utilise heuristic algorithms, as discussed in Section 2.2.9, which cannot guarantee global optimality. For this reason, we propose an improved product targeting formulation and investigate the use of an exact column generation solution framework as a contribution to the product targeting domain. Before going into detail regarding the contributions and advancements made throughout this thesis, a literature review is provided in Chapter 3.

Chapter 3

Literature review

The related work discussed in this chapter focuses on research studies associated with the product targeting optimisation problem found in the literature. The mathematical model formulations investigated in each study are discussed regarding the various business, operational, and channel constraints. Furthermore, the solution methodologies applied to solve the related optimisation problems are also documented to get an idea of the solvers used and the advancements made to generate global or close to global optimal solutions. Finally, a comparative study is performed on the size of the problem instances that were solved in each of the studies to get an idea of the solution capabilities of the proposed algorithms.

3.1 Product targeting optimisation

Nobibon *et al.* (2011) investigated and solved a simplified version of the product targeting problem. The constraints considered included aspects such as the corporate hurdle constraint, budgeting constraint, product limitations, and curbing the number of products assigned to each customer. One of the solution methodologies proposed by Nobibon *et al.* (2011) to solve the preceding product targeting model entailed using a column generation branch-and-price framework. Initially, the Dantzig-Wolfe decomposition theorem was applied to the integer programming (IP) problem to rewrite it into a convex combination of its extreme points. The transposition of the problem resulted in a master problem and a pricing sub-problem (PSP). Then, the column generation algorithm was applied to solve the decomposed form of the basic product targeting problem. Finally, Nobibon *et al.* (2011) proposed the use of a dynamic-programming algorithm in order to solve the pricing problem within pseudopolynomial time. In this research study, eight heuristic algorithms were also investigated to solve the product targeting problem.

These heuristic algorithms were stated to be variations of algorithms already implemented in practice, while some were explicitly designed for the structure of the product targeting problem. The first heuristic algorithm proposed was seen to be a variation of the algorithm presented by Hellinckx (2004). In this heuristic, the assumption was made that the profitability and variable cost were identical for each client in the optimisation problem. All available products were also considered part of the campaign decision process. The second heuristic was again based on the assumptions of the first heuristic but allowed the model to decide which products to commit to instead of forcing it to include all products for assignment to customers. The third procedure successfully solved several exact k-item knapsack problems (E-KKP). The given heuristic was designed to use an approximation algorithm to determine the best product to offer to the selected set of clients. For the fourth algorithm, a next-product-to-buy model proposed by Knott *et al.* (2002) was used in an iterative algorithm to solve the product targeting problem. The fifth heuristic was based on a depth-first branch-and-price algorithm, whereas the sixth heuristic was designed as a truncated call to a mixed integer programming (MIP) solver. A linear programming (LP) rounding heuristic was used to formulate the seventh procedure, while the last heuristic used a tabu-search algorithm to solve the product targeting problem.

In the paper by Nobibon *et al.* (2011), a Dell Optiplex GX620 laptop was utilised with 1.49 GB RAM and 2.8 GHz clock speed to solve the required problem instances. The algorithms were coded in C using Visual Studio C++ 2005 as IDE. Problem instances of up to 10000 customers and 15 products were considered. Some conclusions that were derived from the study conducted by Nobibon *et al.* (2011) were that end users should use exact methods to solve small to medium-sized problems ($\leq$ 2000 customers and 15 products). The previous is suited for business-to-business applications. On the other hand, heuristics should be employed when solving large product targeting problems ($\geq$ 10000 and 15 products), which is more relevant to a business-to-consumer context.

The product targeting formulation proposed by Cohen (2004) followed a similar structure to the models presented by Nobibon *et al.* (2011) but incorporated a few different modeling dynamics by adding an index to the optimisation problem, thus allowing it to account for channel assignment. However, this addition was very limited since it only accounted for the maximum offers that could be assigned to customers via each channel. Channels considered were branches, call centers, and direct mail. In this study, channel complexities such as the best time of day, the probability of right-party contact, the probability of answering, and many other channel-related nuances were not included as part of the study. To solve the presented product targeting formulation, Cohen (2004) mentioned that one could potentially sample a grouping of clients from the exhaustive list of customers and use the sampled list of customers as a representative of the total base. Cohen (2004) identified that this methodology would tend to under-represent customers that fall within the tail end of the distribution and would, therefore, not deliver the desired result. The approach followed was to aggregate customers according to their profit and cost profiles and consider a cluster or a grouping of customers as 1 data point instead of accounting for each customer individually. In doing this, the input data to the problem was significantly simplified, allowing the problem to be solved efficiently. This methodology's limitation is that one might not account for unique customer nuances when clustering customers into various target groups. This methodology will also become increasingly difficult to utilise when considering optimisation constraints focused on selecting and governing personalised customer interactions.

Cohen (2004) considered 11 distinct campaigns, three channels, and approximately 2.5 million customers in the test case that were solved throughout the research study. The 2.5 million customers were clustered using the aggregation approach proposed by Cohen (2004), resulting in a significant reduction in the number of data points being considered. Experiments were conducted with aggregate sizes ranging from 1 to 14731. It was noted that as the size of aggregates was increased, the objective function values started to converge (i.e., increasing the number of aggregates would not significantly improve the solution quality) until it reached the integer relaxed solution. It was, however, proposed that further studies would need to be performed to determine the ideal number of aggregates that would work in a typical banking ecosystem. Cohen (2004) concluded that in implementing the proposed solution methodology, a twofold increase in the incremental campaign profit could be noted when comparing the model outcomes to standard methods currently utilised by the Scotia bank to optimize product targeting.

The contribution made by Delanote *et al.*(2013) to the product targeting problem was to augment the mathematical formulation to account for the possibility of promoting bundled products to customers (cross-selling) as well as the incorporation of multi-channel structures. The mathematical formulation proposed was linked to the legislative framework of Belgian financial institutions. Similar to Nobibon *et al.* (2011) and Cohen (2004), basic constraints, such as the corporate hurdle, campaign budgeting, and the limitation of product offers to customers, were included. Delanote *et al.*(2013) did expand on the channel-related constraints by including aspects such as follow-up capability via multi-channels when an offer has been presented to a customer. Delanote *et al.*(2013) conducted extensive computational experiments and testing on datasets representative of a realistic banking environment. Some strategic applications of the proposed model were also presented as part

of the contributions. The research used a MIP formulation to solve the product targeting problem. Model testing was performed using a Packard Bell laptop iPower GX series equipped with an AMD Athlon X2 processor, 1.90 GHz clock speed, and 4 GB RAM. The version of Cplex utilised to solve the proposed model was 12.2. Problem sizes of up to 15 product offers originating from 4 product categories while utilising four channel dimensions and four customer segments were considered. As part of the testing, Delanote *et al.*(2013) iteratively increased the number of segments considered from 4 to 12 to study the effect of the increase in segment sizes on the computational capacity requirements. Delanote *et al.*(2013) reported that as the segment sizes were increased, the computational time seemed to grow in a strong non-linear fashion. However, given that a segmentation methodology was followed instead of considering each customer as a data point, the proposed MIP model could still comfortably solve the given test instances.

From a strategic perspective, Delanote *et al.*(2013) stated that considering product bundling into the modeling framework significantly increased the probability of the bundled products being taken up compared to only offering separate isolated products to customers. A second aspect of why product bundling seemed desirable was that more products could be offered through a single channel instead of requiring multi-channel costs. Further significant research conducted by Delanote *et al.*(2013) investigated the application of a Benders-based decomposition algorithm in solving the product targeting problem to global optimality or to at least provide a good approximated solution to the problem. It was identified that a multitude of channel complexity constraints, such as multi-number or email addresses associated with single clients and response model inputs to select the best channel option or time of day to contact a customer, were not accounted for in the formulation proposed by Delanote *et al.*(2013).

Lu & Boutilier (2014) proposed a new dynamic segmentation approach for linear programming product targeting problems using a modified column generation formulation as a baseline for the algorithm. Lu & Boutilier (2014) stated that the segmentation was performed using information that falls within categories such as observable, predicted, or derived customer attributes to identify pockets of segments within a customer base. By following the segmentation approach, various customers were treated as indistinguishable from one another, drastically reducing the number of decision variables being considered within the optimisation problem formulation. The segmentation method utilised in this study is known as dynamic cell abstraction (DCA). This method focuses on optimising the product targeting problem for each cell rather than on individual customers. Throughout this research study, constraints related to the campaign and overall budgets were incorporated, and product offering guidelines and channel capacity limitations were taken into consideration to formulate the product targeting problem.

Experiments were performed on datasets ranging from 250000 customers to 10 Million customers while considering 100 possible campaign combinations and channel quantities combinations (Lu & Boutilier, 2014). The data utilised in these test cases were randomly generated using both marketing campaign predictive and value data. The problem instances were solved using commercial LP solvers on high-performance servers containing dual intel 2.6 GHz processors and 244 GB of RAM. From the results obtained, the proposed DCA method could solve the relevant problems to global or near-global optimality within a few seconds. It was concluded that the DCA methodology guaranteed solution quality far superior to the manual or purely statistical segmentation methodologies used in other studies relating to the product targeting problem. Lu & Boutilier (2014) also proposed some future directions which should be explored, including the expansion of the modeling framework to include contact-response models to the mathematical formulation and investigating the possibility of using branch-and-price algorithms to solve the product targeting problem.

The work conducted by Smaili & Hachimi (2021) focused on two stages of the product targeting process: customer segmentation and customer targeting. First, a machine learning algorithm (k-means clustering) was employed to predict and cluster customer behavior toward product offerings,

where the preceding data was fed as input into the customer targeting strategy. The segmentation methodology used was based on a method that is known as the Recency Frequency Monetary (RFM) model. According to Smaili & Hachimi (2021), the RFM method is widely used in retail banking. After applying RFM and k-means clustering to predict the various clusters, Smaili & Hachimi (2021) proposed a MIP customer targeting model to generate an optimal marketing strategy. The MIP model considered business and operational constraints such as the corporate-hurdle rate, campaign budgets, product limitations, and minimum quantity commitment.

To solve the MIP optimisation problem, Smaili & Hachimi (2021) utilised a method known as hybridization of meta-heuristics. In layman's terms, it translates to a technique where a set of meta-heuristics are employed sequentially to solve the product targeting problem . The output of the previous heuristic forms the input of the following heuristic. In addition, meta-heuristics were also utilised in parallel to solve the product targeting problem. In this way, the meta-heuristics were each launched at a point in time within the search space while interacting with one another to compute the optimal solution. The four heuristic classes which were used are known as low-level relay hybrid (LRH), low-level teamwork hybrid (LTH), high-level relay hybrid (HRH), and high-level teamwork hybrid (HTH). The data utilised to solve the given problem instances were randomly generated by Nobibon *et al.* (2011). Smaili & Hachimi (2021) considered small (100 customers), medium (1000 customers), and large (10000 customers) problem instances to test the capability of the solution framework proposed. The number of products considered ranged from 5 – 10 in various combinations. Results discussed by Smaili & Hachimi (2021) were quite limited. However, it was concluded that the meta-heuristic hybridization algorithms allowed for better solutions compared to other literature studies.

Cetin & Alabas-Uslu (2017) proposed heuristic approaches to solve the product targeting problem with a specific focus on mathematical programming. The approaches initially identified the list of products that would participate in a campaign by applying heuristic rules, where the identified products were optimally distributed to customers. Cetin & Alabas-Uslu (2017) reported that the proposed heuristic algorithms generated superior results compared to existing solution algorithms. Furthermore, the applicability of the algorithms was tested on very large-sized test instances to gauge the potential to implement the algorithms practically. The product targeting problem formulation considered in this study was similar to that proposed by Nobibon *et al.* (2011). This formulation considers the corporate hurdle constraint, budgeting constraint, product limitation, and maximum product offers per customer.

The product targeting formulation presented by Cetin & Alabas-Uslu (2017) was divided into two phases. The first phase was a new linear programming formulation focused on determining which products should form part of the campaigning process. Once the products were identified, the product targeting model was reduced to a generalised assignment problem. In the second phase of the proposed solution methodology, another optimisation model was solved to distribute the selected products identified in phase 1 to the relevant customers. The two modeling phases were connected via a heuristic rule set. The proposed heuristic algorithms were tested using data sets from literature such as that utilised by Nobibon *et al.* (2011). Datasets containing 10000 customers and 15 products were considered during the testing phases. Cetin & Alabas-Uslu (2017) reported that given the limited research available on the topic of product targeting optimisation, the proposed heuristic algorithms were compared to those presented by Nobibon *et al.* (2011). After performing a comparative study, it was concluded that the heuristic algorithms designed by Cetin & Alabas-Uslu (2017) significantly outperformed existing heuristic algorithms concerning solution quality and compute time. Some future work proposed by Cetin & Alabas-Uslu (2017) was to add additional business constraints to the product targeting formulation to gauge the effectiveness and capability of the proposed heuristic algorithms to solve more complex product targeting formulations.

The focus of the study conducted by Oliveira *et al.* (2015) was to optimise the number of posi-

tive product purchases/take-ups by customers while minimising the operational cost associated with the direct marketing campaigning process. In this study, a hybrid heuristic algorithm was proposed, with its core being based on the greedy randomized adaptive search procedures as well as the general variable neighborhood search theories. Furthermore, Oliveira *et al.* (2015) attempted to apply the preceding heuristics in order to solve the targeted offers direct marketing campaigns (TODMC) problem as proposed by Nobibon *et al.* (2011).

Oliveira *et al.* (2015) performed the test on an OPTIPLEX 9010 Intel Core i7-3770, 3.40 x 8 GHz computer, containing 32 GB of RAM. The model was developed and compiled with g++ 4.6.3 using the Eclipse Kepler release. As part of the testing, various problem instances were considered, with sizes ranging from small (100) to large (10000). Product sets considered varied between a range from 5 to 15 products. After conducting comparative studies on the outcomes of the tests performed by Nobibon *et al.* (2011), it was concluded that the heuristic algorithm proposed by Oliveira *et al.* (2015) was able to obtain lower or equal optimality gaps to that of Nobibon *et al.* (2011). Some further research avenues proposed by Oliveira *et al.* (2015) entailed the addition of new constraints, such as variable costs when offering large amounts of products to customers. Potentially also reducing the cost at which a product should be offered as the number of customers receiving the offer increases. Oliveira *et al.* (2015) stated that the preceding would be valuable additions to the product targeting formulations to consider as the mentioned dynamics are the type of nuances that occur in real live direct marketing settings.

Coelho *et al.* (2017) conducted a study focused on optimising a bi-objective function product targeting problem using heuristic algorithms to solve the given problem. The bi-objective function was designed to maximise the expected profit generated by direct marketing campaigns while also increasing the risk-adjusted return computed using the reward-to-variability ratio known as the Sharpe ratio. Coelho *et al.* (2017) opted for the use of heuristic algorithms to solve the problem due to the large volume of data as well as the combinatorial nature of the problem. The heuristic algorithm designed to solve the product targeting optimisation problem is known as a Greedy randomized neighborhood structure. The proposed algorithm included the dynamics of the Greedy randomized constructive techniques and the neighborhood exploration strategy. In addition, the product targeting problem considered in this research paper took constraints such as the minimum desired profit, budgeting, product offer availability, and customer requirements into consideration.

The meta-heuristic algorithm proposed by Coelho *et al.* (2017) was developed using C++ within the OptFrame 2.2. The computational experiments conducted by Coelho *et al.* (2017) were performed using an OPTIPLEX 9010 Intel Core i7-3770, 3.40x8 GHZ computer with installed 32 GB of RAM. The operating system utilised was Ubuntu 14.04, and the code was compiled by g++ 4.8.4 while using the Eclipse Kepler release. The dataset used for testing was derived from the instances proposed by Nobibon *et al.* (2011) with problem instances ranging from 300 – 10000 clients and product offerings consisting of 5, 10, and 15 products in different combinations. Coelho *et al.* (2017) focused on testing the characteristics obtained from the Pareto Fronts in solving the preceding test cases. The Sharpe ratio index proposed also managed the search for low-risk direct marketing campaigns while regulating a trade-off between the groups of clients to which offers were presented and the total campaign profit generated. It was also stated that the proposed heuristic algorithms used multi-core processing technology to improve the computability of the proposed algorithms.

In the research paper compiled by Bhaskar *et al.* (2009), the problem of selecting the optimal list of customers to which cross-sell campaigns should be proposed within the retail banking environment was investigated. Bhaskar *et al.* (2009) stated that to generate such an optimised customer list, various aspects such as expected volume, response propensity, and expected profit from customers should be accounted for. To solve the preceding product targeting problem, Bhaskar *et al.* (2009) proposed a fuzzy mathematical programming technique. To apply the mentioned solution methodology, Bhaskar *et al.* (2009) represented both the constraints and the imprecise parame-

ters as triangular fuzzy numbers. To address the issue of computational complexity, Bhaskar *et al.* (2009) proposed a group-level formulation of the problem. This research study considered and solved a real-life cross-sell problem from a bank. The objective function considered for the product targeting formulation was dependent on three variables, with the first being the response propensity of a customer to a given product offer, the second was the potential profit gain in the event of a positive response, and the third was focused on the risk of default. The preceding parameters were predicted using historical data to formulate the optimisation problem under consideration. Cost accounted for in the product targeting formulation included targeting cost and operational expenses. The two main constraints considered in this research study were the campaign budget and business asset growth targets. To reduce the problem's computational complexity, Bhaskar *et al.* (2009) also exploited a grouping/ segmentation methodology where customers were grouped according to their profit score and response probabilities.

The problem instance, which was solved using the proposed methodology elaborated on by Bhaskar *et al.* (2009), consisted of 184115 customers and 3 product offers. Following the grouping methodology, the customer base was divided into 9685 populated groups. Bhaskar *et al.* (2009) solved the preceding problem instance and utilised the results obtained to compare the performance of the fuzzy formulation with that of a crisp formulation. Future research proposed by Bhaskar *et al.* (2009) included the investigation of a better segmentation methodology as the one implemented would increase in complexity as the number of products considered were to increase. Other study directions were also proposed, specifically focusing on aspects such as how the fuzzy numbers were derived or how to capture uncertainty in parameters relating to the proposed solution methodology.

The work proposed by Nair & Tarasewich (2003) focused on generating an optimal design of a series of promotional offers to customers via regular mail. The offers included gifts, special services, or discounts on selected items. The objective function that Nair & Tarasewich (2003) set out to maximise was the multiple purchases of the interested customer base over time. The solution algorithm proposed by Nair & Tarasewich (2003) entailed using a Genetic algorithm-based heuristic specifically designed to derive acceptable promotion offers for selected customers. The solution methodology was tested using actual data. The reason why the study conducted by Nair & Tarasewich (2003) specifically focused on promotional offers was as a result of research that indicated that sales promotions stimulated excellent target market responses when compared to offers without using promotions. In the presented study, constraints accounting for utility value, multiparty loss, selection of one offer, minimum duration between offers, and analyst characteristics were accounted for. The results indicated that the Genetic-algorithm-based heuristic could generate reasonable solutions within a short period. The study conducted by Nair & Tarasewich (2003) proposed quite a different integer programming formulation to that presented by the other research studies discussed in Chapter 3, with a specific focus on generating a promotion list over time. For this thesis, the focus was instead set on the solution methodology applied by Nair & Tarasewich (2003) due to the applicability of the problem formulation to the direct marketing domain rather than focusing on the IP formulation constraints utilised to govern the model outcomes.

3.2 Related optimisation problems

Throughout the literature, complex NP-hard problems could be identified, similar to the product targeting optimisation problem investigated in this thesis. These problems entailed the stock-keeping unit (SKU) problem (Ma & Fildes, 2017), the general assignment problem (GAP) problem (Savelsbergh, 1997) as well as the opportunistic maintenance planning problem (Friberg, 2015). In each of the preceding instances, the problems entailed identifying an exhaustive list of decision variables to maximise or minimise their relevant objective functions. By investigating the solution algorithms researchers apply to solve the preceding optimisation problems, one could identify potential solution algorithms that could be applied in this research study to solve the IP formulation product targeting problem proposed in Chapter 4.

Ma & Fildes (2017) proposed a store stock-keeping-unit model that maximises category multi-period profit. A two-step approach was proposed to manage promotion decisions' interactive and dynamic implications. The presented methodology also allowed for accurate forecasts of profit and sales levels while generating efficient promotion plans to maximise long-term profitability. The algorithm implemented by Ma & Fildes (2017) to solve the preceding problem is known as the Genetic algorithm. It was utilised to scour the search horizon for a set of decisions that would be able to maximise category profit. No in-depth analysis was conducted on the constraints and modeling structure of the binary integer programming formulation proposed for the promotional campaign model, as there were no strong resemblances to the product targeting formulation investigated in this thesis. However, the topic of interest was the complexity of the problem to be solved and the Genetic algorithm approach followed by Ma & Fildes (2017) to generate acceptable solutions to the problem instances considered. The Genetic algorithm was successfully applied to solve the promotional campaign problem; however, it was identified by Ma & Fildes (2017) that the given algorithm could have been more time-consuming when applied to large-scale problem instances. Therefore, it was proposed by Ma & Fildes (2017) to investigate the use of an alternative heuristic algorithm to speed up the process of computing an optimal solution when conducting future research.

Savelsbergh (1997) proposed a branch-and-price algorithm to solve the GAP problem using column generation and branch-and-bound approaches. The GAP problem is concerned with maximising the profit attained by assigning jobs to agents while satisfying the relevant capacity restrictions prescribed by the end user. The proposed formulation comprised an objective function and two constraints. In work proposed by Savelsbergh (1997), some investigations were conducted to determine the effect of varying the pricing problem solution algorithms and branching strategies on the overall solution quality. It was concluded by Savelsbergh (1997) that the proposed branch-and-price strategy utilised in the cited research paper outperformed available literature algorithms (Lagrangian dual algorithm) for GAP problems with small (n: jobs)/(m: agents) ratios (i.e.,$n/m \leq 5$). Furthermore, for GAP instances where the n/m ratios exceeded 5, the mentioned Lagrangian dual algorithm outperformed the branch-and-price algorithm.

Friberg (2015) also proposed a branch-and-price algorithm to solve the opportunistic maintenance planning problem related to the RM12 aircraft engine. In this master thesis, Friberg (2015) alluded that the basic branch-and-price algorithm has many aspects that could be enhanced to improve algorithmic performance. This research study was focused on exploiting these possible improvements and the effect of the preceding on the solution quality. Friberg (2015) also touched on aspects of the heading-in, yo-yo, and tailing-off effects, as discussed in Chapter 5 of this thesis. Important points noted by Friberg (2015) during the concluding remarks of the master thesis was that to improve the performance of the branch-and-price algorithm, the focus should be set on designing a problem-specific heuristic capable of generating reasonable initial incumbent solutions to the reduced master problem. In addition, it was also mentioned that one should focus on designing heuristics that could speed up the solution process of the pricing problem to generate columns with negative reduced costs significantly faster when compared to using commercial MIP solvers. Exploring the possibility of exploiting parallel computing within the sub-problem domain was also mentioned to reduce the solution time required for the overall branch-and-price algorithm.

Similar to Friberg (2015), the work presented by Pessoa *et al.* (2017) also investigated stabilisation techniques that were developed to improve the performance of column generation algorithms. In the mentioned research paper, techniques such as using the proximity of a stability center through implementing penalty functions were discussed. Other techniques linked to dual variable smoothing were also investigated, with the last method termed the centralized prizes approach. The work conducted by Pessoa *et al.* (2017) was, however, not explicitly focused on the product targeting problem but on enhancing the column generation algorithmic performance. The work completed by Friberg (2015) and Pessoa *et al.* (2017) eluded to the fact that when implementing algorithms such

as Dantzig-Wolfe decomposition and column generation to solve NP-hard optimisation problems, the standard algorithm formulations will generally not deliver the desired performance improvements on its own. It is required that, for example, the column generation algorithm be enhanced by adding algorithmic improvements such as stabilisation techniques in order to obtain the desired computational outcomes.

3.3 Concluding remarks

The initial review provides insights into the advancements made to the product targeting optimisation problem throughout the literature. The majority of the researchers studied a reasonably simplistic version of the product targeting problem formulation that only accounted for basic product and customer-related nuances. Individual researchers expanded the basic formulation to include some channel selection capabilities, which were seen to be very limited. It was also noted that in most studies, heuristic algorithms or segmentation approaches were proposed to solve the product targeting problem. In this thesis, a IP formulation is proposed taking into account a multitude of additional business, operational, and channel constraints to closely align the problem to real-life scenarios while improving on the existing product targeting formulations and solution methodologies. After enhancing the product targeting formulation, guidance is taken from the studies investigated throughout Section 3.2 to propose a column generation framework to solve the IP formulation product targeting problem. The contributions proposed in this thesis enhance the work presented by Nobibon *et al.* (2011). Many mathematical alterations and algorithmic improvements to the already existing product targeting formulation were proposed to increase the size of the product targeting problems that could be solved successfully. Details regarding contributions are discussed in Chapters 4 and 6 below. Even tough larger problems can be solved, it is evident that the column generation algorithm proposed in this thesis struggles from a tailing-off effect which will be discussed in more detail in Chapter 5. The resulting effect is a lot of the problems struggling to reach global optimality and only producing solutions that are close to global optimality (gap ranging between 2% – 10%).

Chapter 4

Product targeting model formulation

4.1 General notation

This chapter presents a IP mathematical formulation for the product targeting problem. The focus is initially on discussing the mathematical details related to a basic IP formulation; whereafter the IP formulation is introduced. Then, insights are provided regarding the dynamics of each constraint contained within the basic and formulations and the complexity associated with each. However, before discussing these mathematical formulations, some general notations are introduced.

Let $\mathcal{J}$ denote the index set of the number of product offers being incorporated into the product targeting problem, with $j \in \mathcal{J}$ denoting a specific product offer within the collection. The grouping of customers to which a product could be offered is denoted by index set $\mathcal{I}$, with $i \in \mathcal{I}$ referring to a specific customer within the group of customers being considered. To account for the number of channels available to contact customer i for product j, we introduce index set $\mathcal{U}$ with $u \in \mathcal{U}$ representing a unique channel within the collection of channels which could be utilised as the communication medium.

A time index set $\mathcal{T}$ is added to the product targeting problem to account for the best time $t \in \mathcal{T}$ at which a customer should be called when voice is selected as the channel of preference. Let $\mathcal{Q}$ denote the collection of phone numbers and email addresses linked to the collection of customers $\mathcal{I}$. The subset $\mathcal{Q}(i, u) \subseteq \mathcal{Q}$ denotes the various numbers and email addresses that belong to a unique customer $i \in \mathcal{I}$.

Another index set introduced into the mathematical framework is represented by $\mathcal{C}$, with $\mathcal{C}$ alluding to the cross-sell options available within a given product targeting problem. The subset $\mathcal{C}(i, j) \subseteq \mathcal{C}$ does, however, delineate the cross-sell options available to each customer $i \in \mathcal{I}$ after being offered product $j \in \mathcal{J}$.

4.2 Baseline IP formulation

The product targeting problem has multiple mathematical variations, as deliberated in Chapter 3. However, as seen throughout the literature study, there is a general trend of a standard grouping of business and operational constraints included as part of the product targeting problems investigated by the various researchers. The preceding grouping of constraints constitutes the baseline model being considered in this thesis. It is, however, essential to understand that throughout the literature, there was already some work done by various researchers to improve the baseline model, which will be presented in Section 4.2. Refer to Chapter 3 for the summary provided. Initially presenting the mathematical formulation of the baseline model creates the context for the reader regarding the general departure point utilised by various researchers when trying to improve or expand on the product targeting problem. In the subsequent sections, 4.2.1 and 4.2.2, a detailed overview of the baseline mathematical formulation and the reasoning associated with the inclusion of the relevant

constraints into the model will be provided.

4.2.1 Model objective

In (4.1), the objective function for the baseline IP formulation is provided. This objective function incorporates the basics related to the product targeting problem with the value $c_{ij}^{(v)}$ denoting the variable cost incurred by a financial institution when offering products j to potential customers i. We denote the parameter p_{ij} as the expected return on investment obtained by the financial institution when offering products j to customers i. The value of p_{ij} is calculated as the potential income r_{ij} which could be obtained from each product j, multiplied by the probability of customer i to take-up product j. The preceding is represented by T_{ij} ($p_{ij} = r_{ij}T_{ij}$). For ease of representation, p_{ij} is used in the subsequent mathematical derivations. The total profit obtained from the campaign selection process is calculated by subtracting $c_{ij}^{(v)}$ from p_{ij}. However, during the preceding calculation, only the indices where decision variable $x_{ij} \in \{0, 1\}$ is assigned a value of 1 will affect the objective function calculation. When decision variable x_{ij} is assigned a value of 1, it means that product j has been queued to be offered to customer i. Thus the associated cost and income will need to be taken into consideration. The contrary is true when the decision variable x_{ij} is assigned a value of 0. Then no product j is considered for customer i.

The product targeting problem objective function also considers any fixed cost associated with setting up a campaign for product j and allowing the product to be offered to the available client base. The fixed cost is represented by $c_j^{(f)}$. It is, however, imperative to note that the fixed cost associated with product j will only be considered in the optimisation solution if the product under consideration has been assigned to at least one customer i. If a specific product has not been assigned to any customer i, the decision variable $y_j \in \{0, 1\}$ will be assigned a value of 0 for the associated index j. When the decision variable has been assigned a value of 0, it will, in turn, prevent the affected product from forming part of the optimisation solution. The opposite is true when the decision variable y_j is assigned a value of 1. In this case, product j will be allowed to enter the decisioning process.

$$(B_1) \quad \max \quad \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} (p_{ij} - c_{ij}^{(v)}) x_{ij} - \sum_{j \in \mathcal{J}} c_j^{(f)} y_j \tag{4.1}$$

The ultimate aim of the baseline objective function (B_1) is to maximise the expected profit by allowing the model to decide which products j should be assigned to customers i, by means of allocating binary values to decision variables x_{ij} and y_j. In this version of the objective function, no complex aspects such as channel type, time of contact, or cross-sell options were considered. Once the baseline has been established, the optimisation model complexity can be increased significantly to align the model to real business and operational outcomes, which is the end state of this thesis.

4.2.2 Model constraints

The first constraint included in the basic IP formulation model is constraint set (4.2), the corporate hurdle-rate constraint. Constraint set (4.2) is responsible for ensuring that the return on investment (ROI) for the campaigns that form part of the decision framework is at least R. Parameter R is calculated as $(1 + r)$ with r representing a fraction greater than 0. The preceding means that when the campaign expenses (right-hand side (RHS) of constraint set (4.2)) are multiplied with R, it is inflated by a predefined percentage which is provided as an input to the mathematical model. The campaign expenses constitute fixed and variable costs associated with the offers j being made per customer i. Suppose a specific customer or product is excluded from the decision process. In that case, the associated terms in constraint set (4.2) will default to a value of 0, and the constraint will remain true. The decision variables x_{ij} and y_j will only be allowed to take on values of 1 within the confines of this constraint, ensuring that the overall ROI achieved during the product targeting process remains greater than or equal to the RHS of constraint set (4.2). By varying parameter R,

the end user can manipulate the number of customers and products being committed in the decision process and consequently increase or decrease the percentage profit being made.

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} p_{ij}x_{ij} \geq R(\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}^{(v)}x_{ij} + \sum_{j\in\mathcal{J}} c_j^{(f)}y_j) \tag{4.2}$$

In the campaign environment, each product house has an annual budget to which it must adhere. If the various product houses do not adhere to their budget requirements, it could lead to unwanted outcomes for the financial institution. In order to account for this business requirement, constraint set (4.3) is added to the basic IP formulation. Parameter B_j delineates the budget linked to each product j in the campaign process. On the left-hand side (LHS) of constraint set (4.3), the variable cost associated with the various campaigns is taken into consideration by means of multiplying it with the decision variable x_{ij}. After that, it is set to be less than or equal to B_j to meet the budget requirements. Note that the variable cost for each customer i and product j is only considered when decision variable x_{ij} is assigned a value of 1. If a value of 0 is assigned to x_{ij} then for the specific index i and j, no expense will be considered, and it will not influence the overarching budget constraint (4.3).

$$\sum_{i\in\mathcal{I}} c_{ij}^{(v)}x_{ij} \leq B_j, \quad j \in \mathcal{J} \tag{4.3}$$

To govern the number of offers assigned to each customer i, constraint set (4.4) is added to the model. This constraint takes the summation of all the products j linked to a selected customer i and ensures that it is less than or equal to 1. The RHS of this constraint can be updated to a value greater than one if the campaign setup allows a customer to receive multiple offers. However, for the study under consideration, we only allow the mathematical model to assign one product j to each customer i if the customer is selected to form part of the campaign setup. If a customer is excluded from the product targeting problem, the summation of decision variable $\sum_{j\in\mathcal{J}} x_{ij}$ for the affected customer will equate to 0. The RHS has been assigned a value of 1 to allow the model to provide the end user with the best possible product for each customer, which would increase the net benefit received by the financial institution.

$$\sum_{j\in\mathcal{J}} x_{ij} \leq 1, \quad i \in \mathcal{I} \tag{4.4}$$

The constraint above governs the number of products the model can assign to each customer. However, as part of the product targeting problem, a limit on the number of customers assigned to each product should also be considered. Constraint sets (4.5) and (4.6) are added to the optimisation model to enforce these limitations. Constraint set (4.5) provides a lower bound on the number of customers assigned to an associated product j. The lower bound is denoted by parameter $l_j^{(l)}$ and it is multiplied by decision variable y_j. The reason why the lower bound is multiplied by y_j is to ensure that when a product is excluded from the decision process (i.e., $y_j = 0$), the constraint will still hold true and henceforth not lead to model infeasibility issues. The upper bound for the number of customers assigned to product j is managed by constraint set (4.6) with parameter $l_j^{(u)}$ providing the upper bound as input. Similar to constraint set (4.5), the upper bound $l_j^{(u)}$ is also multiplied with the decision variable y_j in order to account for products not forming part of the decision process. If $y_j = 0$, then the upper bound will default to 0 and, as a result, ensure that for the affected product j, no customer allocation can take place.

$$\sum_{i\in\mathcal{I}} x_{ij} \geq l_j^{(l)}y_j, \quad j \in \mathcal{J} \tag{4.5}$$

$$\sum_{i\in\mathcal{I}} x_{ij} \leq l_j^{(u)}y_j, \quad j \in \mathcal{J} \tag{4.6}$$

The final constraint included in the baseline IP formulation is the integrality constraint, represented by constraint set (4.7). This constraint is included in the mentioned model to allow decision variables x_{ij} and y_j only to take on binary values of either 0 or 1.

$$y_j, x_{ij} \in \{0, 1\}, \quad i \in \mathcal{I}, j \in \mathcal{J} \tag{4.7}$$

Constraint sets (4.1) – (4.7) represent the collection of business and operational guidelines and limitations imposed on the baseline IP model formulation. In the subsequent section, the baseline model's complexity and relevance are expanded by considering dynamics such as channel of interaction, time of contact, cross-sell, and various other intricacies. These additional factors also influence the already defined constraints as they will need to be adapted to fit into the newly defined model framework. Throughout the literature, slight variations of the provided baseline model might exist. However, the proposed model is a good representation of ground truth.

4.3 Novel IP formulation

The baseline IP model formulation only touches on specific aspects of the overarching product targeting problem and does not go into detail on critical business and operational constraints. The previous is why this study first focuses on expanding the baseline model by proposing a product targeting IP model formulation. The proposed mathematical model incorporates aspects such as channel of communication (limited to voice, SMS, or email), marketing consent, recency, time of day to contact, and various other critical topics into the model in order to ensure that it is as closely aligned to real business and operational guidelines as possible. To be exact, as part of the IP model formulation contributions, a total of 12 additional constraints and three decision variables were introduced, and the dimensions of the problem were expanded with three additional indices. Throughout Sections 4.3.1 and 4.3.2, information will be provided regarding the preceding, focusing on the intricacies of the newly defined model objective function and its associated constraints.

4.3.1 Model objective

The IP formulation objective function is represented by (4.8) – (4.12). Similar to the objective function discussed in Section 4.2.1, this objective function aims to maximize total profit, but some additional dimensions have been added. In (4.8), the term P denotes the total monetary gain obtained by the financial institution as a result of committing products to customers in an optimal fashion. Term CH is introduced into the objective function to allow the model to select the best possible mobile number or email address for a specific customer if one customer might have multiple phone numbers or email addresses. Term TM is used in the objective function to allow the mathematical model to select the best time of day to call a customer if voice was selected as the preferred contact channel. Lastly, the term CR is added to the objective function to account for cross-selling opportunities for selected products. The following sections will provide a detailed breakdown of each term.

$$(M_1) \quad \max \quad (P + CH + TM + CR) \tag{4.8}$$

Similar to (4.1), in (4.9), the monetary gain obtained by the financial institution is calculated by subtracting the variable cost $c_{iju}^{(v)}$ from the potential income p_{iju}, where after it is multiplied with decision variable $x_{iju} \in \{0, 1\}$. Decision variable x_{iju} determines which customer i should be offered product j. In (4.9), the model also accounts for the fixed cost associated with offering product j to customer i by incorporating parameter $c_j^{(f)}$ and decision variable $y_j \in \{0, 1\}$. In order to introduce channel complexity to the product targeting problem, the objective function parameters as well as decision variable x_{iju} noted in (4.9) were expanded to include index u. Index u, as previously mentioned, is representative of the possible channels available which could be utilised to contact customers i in order to sell products j. In addition to the preceding, r_{iju} is added to (4.9) to account for the probability of reaching a customer on a specific channel (i.e., probability of answer).

Parameter r_{iju} is multiplied by the monetary gain term to adjust the expected profit according to the chances of obtaining a successful answer. If there is a low probability of an answer, the expected monetary gain will be penalized, whereas the opposite is true when a high answer probability is registered. The optimisation model would rather choose a customer with a high answer rate as it will improve the chances of a successful sale. By only considering the probability of getting hold of a customer, one assumes that when there is a customer with a high probability of an answer, it will be the intended customer at the other end of the line when an answer is achieved. However, reaching a customer does not imply that it will be the correct customer. A customer could have a low probability of an answer, but there could be a high certainty that when getting an answer, it would be the correct customer and vice versa. Therefore, one would also need to take the probability of getting hold of the right party contact (RPC) into account in combination with the probability of an answer to have a balanced and representative model. The preceding will only be considered in (4.10) and will not be included as part of (4.9). In (4.9), only the dynamics associated with the probability of an answer are accounted for.

If a customer has provided the financial institution with a preferred medium of contact, parameter $c_{iju}^{(p)}$ is added to (4.9) in order to augment r_{iju}. Suppose a customer has selected a single channel as the preferred channel. In that case, parameter $c_{iju}^{(p)}$ will take on a value of 1 for the associated index u, with the rest of the channels being assigned a value of 0. By adding r_{iju} and $c_{iju}^{(p)}$ together, it automatically forces the model to choose the term with the highest calculated value, which in this case will be the preferred channel of contact as selected by the customer. A customer could also have multiple preferred channels, with each of them having a value of 1 assigned to $c_{iju}^{(p)}$. In the event of multiple preferred channels, the channel with the highest probability of answering will take precedence over the other.

Another weighting parameter, $c_{iju}^{(s)}$, is added to (4.9) to allow products that have cross-sell opportunities to take priority over those products that do not have cross-sell options. Thus, there is a possibility that more than one product j could be offered to customer i if a cross-sell option is available. The preceding will, in turn, be more profitable to the financial institution in the event of a successful sale. Parameter $c_{iju}^{(s)}$ is added to the multiplication term in (4.9) to force the objective function to prefer offers when there are cross-selling opportunities. As a final addition to the monetary gain function, parameter m_1 is added to the multiplication term, which allows the end user to assign an importance weighting to term P. When the end user wants to focus more on answer rates, preferred channels, and cross-sell options, a large constant value would be assigned to m_1. If, however, these terms are not the deciding factors for the end user, then a smaller weighting could be assigned to m_1. A smaller weighting would result in (4.10) – (4.12), for example, taking precedence when considering the overall objective function of (4.8), and focus would be placed on aspects such as RPC rates instead.

$$P = \sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{I}} \sum_{u\in\mathcal{U}} ((p_{iju} - c_{iju}^{(v)})x_{iju})(r_{iju} + c_{iju}^{(p)} + c_{iju}^{(s)})m_1 - \sum_{j\in\mathcal{J}} c_j^{(f)} y_j \tag{4.9}$$

As noted in the above section, term CH is added to the product targeting objective function in order to allow the model the capability to decide which cellphone number or email address is the best to use if customer i has multiple phone numbers or email addresses. In order to achieve this capability, we introduce parameter $c_{ijuq}^{(q)}$ which denotes the probability of RPC for each phone number or email address associated with a given customer i. The model would be prone to select the number or email address that has the highest RPC rate for the selected customer i, using parameter $c_{ijuq}^{(q)}$. Note that the selection between whether email or phone number (voice or SMS) should be used as a contact method to contact the customer will depend on the type of channel selection the model performs. The preceding selection will, however, be determined by the assignment of x_{iju}. Once the assignment of x_{iju} has been finalised, binary decision variable $c_{ijuq}^{(d)} \in \{0, 1\}$ in (4.10) is utilised to allow the model to select the best possible phone number or email address, where x_{iju} has been assigned a value of 1. In order to perform this selection, $c_{ijuq}^{(d)}$ is assigned a value of 1 when a specific

number or email address associated with x_{iju} is chosen. When no selection is performed, a value of 0 is allocated to $c^{(d)}_{ijuq}$, excluding the influence of the remaining channels from the model. Similar to (4.9), a weighting parameter m_2 is added to (4.10) in order to allow the end user to specify the influence of (4.10) on the overall objective function by assigning a constant value to m_2.

$$CH = \sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{I}} \sum_{u\in\mathcal{U}} \sum_{q\in\mathcal{Q}_{iu}} (c^{(q)}_{ijuq} c^{(d)}_{ijuq}) m_2 \tag{4.10}$$

Suppose decision variable x_{iju} has been utilised to select voice ($u = 1$) as the best channel of contacting customer i and decision variable $c^{(d)}_{ijuq}$ has confirmed the best number to use for the said channel, we need to introduce another dimension into the optimisation model. This dimension refers to the capability of the model to select the best time t to call a customer i on the selected phone number q. This ability is enabled by incorporating (4.11) into the objective function. Input parameter $c^{(t)}_{iuqt}$ in (4.11) denotes the probability of answer at different periods of the day for a selected phone number q. For the mathematical model considered in this thesis, we only incorporated three time periods (morning, afternoon, and evening) as selection options within the model. To perform the time selection, binary decision variable $h_{jiuqt} \in \{0, 1\}$ is incorporated into (4.11). In order for the decision variable h_{jiuqt} to select a desired time t, a value of 1 will be assigned to the relevant index of the decision variable. When a time t is not considered in the model, a value of 0 will be assigned to h_{jiuqt}.

$$TM = \sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{I}} \sum_{u=1} \sum_{q\in\mathcal{Q}_{iu}} \sum_{t\in\mathcal{T}} c^{(t)}_{iuqt} h_{jiuqt} \tag{4.11}$$

The last component in the product targeting objective function is CR. As mentioned, this parameter refers to the capability of the model to include cross-sell options c when offering certain products j to customers i. Only selected products would have cross-sell options, and not all available products may be combined in a cross-sell combination. These cross-sell combinations are taken as input to the optimisation model and are encompassed by index $\mathcal{C}_{ij}$. Parameter $c^{(r)}_{ijc}$ denotes the additional financial gain which could be realized when a cross-sell option is selected. Binary decision variable $o_{ijc} \in \{0, 1\}$ is multiplied with parameter $c^{(r)}_{ijc}$ in order to exclude the financial gain of cross-selling options if no such option exists or if it is not the desired selection to perform for the customer under consideration. Exclusion of the parameter mentioned above is achieved when o_{ijc} is assigned a value of 0. However, when a cross-sell option is selected to form part of the product offering, decision variable o_{ijc} takes on a value of 1, and the financial gain is realised.

$$CR = \sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{I}} \sum_{c\in\mathcal{C}_{ij}} c^{(r)}_{ijc} o_{ijc} \tag{4.12}$$

To compile the objective function for the product targeting problem, (4.8) – (4.12) are lumped into a maximisation function. By incorporating the aforementioned constraints into the objective function, the problem is focused on maximising the financial institution's monetary gain and selecting the best conversation at the right time through the correct channel to improve customer satisfaction. As a result, the complexity of the objective function for the product targeting problem is increased substantially compared to the baseline model noted in Section 4.2. In the subsequent section, a detailed outline will be provided regarding the IP formulation constraints and the reasoning behind each inclusion.

4.3.2 Model constraints

As part of the product targeting IP formulation, constraint sets (4.13) – (4.30) are added to the mathematical model with the first five constraint sets, (4.13) – (4.17), corresponding to modified versions of constraints that were already discussed in Section 4.2.2. The only enhancements to the aforementioned constraints entail the addition of index u to incorporate channel decisioning complexity and the addition of recency tracking capability. However, the preceding improvements will

be elaborated on when delving deeper into the reasoning and complexity behind each constraint. Although constraint sets (4.13) – (4.17) were augmented to incorporate the above aspects, the novelty of the proposed product targeting IP formulation originates rather from the addition of constraint sets (4.18) – (4.30) as the literature relating to the mentioned constraints and the convolutedness associated to the inclusion of these constraints to the mathematical model was very limited. Therefore, initially, a detailed discussion is provided related to constraint sets (4.13) – (4.17), where after we investigate constraint sets (4.18) – (4.30).

Consider constraint set (4.13). Like constraint set (4.2), the corporate hurdle constraint for the product targeting problem is represented by constraint set (4.13). No additional parameters or variables were added to the corporate hurdle constraint for the product targeting problem. Thus the explanation provided in Section 4.2.2 concerning constraint set (4.2) still holds. The only added complexity is the addition of index u, which allows decision variable x_{iju} to propose that product j be marketed to customer i via channel u. In constraint set (4.13), all the variables have been shifted to the left, and the right-hand side has been set equal to or greater than 0. Constraint set (4.13) still has the same mathematical relevance as was noted in constraint set (4.2). However, the preceding transformation was required to reduce complexity when performing the dual problem derivation in Section 6.3.3. Note that the influence of decision variable y_j and input parameter $c_j^{(f)}$ is independent of channel and for this reason index u is omitted from the multiplication term $(c_j^{(f)} y_j)$ in constraint set (4.13).

$$\sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}}\sum_{u\in\mathcal{U}} p_{iju}x_{iju} - R(\sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}}\sum_{u\in\mathcal{U}} c_{iju}^{(v)}x_{iju} + \sum_{j\in\mathcal{J}} c_j^{(f)}y_j) \geq 0 \tag{4.13}$$

When considering the budgeting constraint set (4.14) implemented in the product targeting IP formulation, it should be noted that no significant alterations have been made to this constraint. The right-hand side term B_j has remained unchanged while the only change to the left had term is the addition of index u to decision variable x_{iju} in order to account for channel selection. Similar to constraint set (4.3), constraint set (4.14) is implemented in order to ensure that the number of customers i contacted through channel u does not exceed the prescribed campaign budget for each product j which was set out by the various business units.

$$\sum_{i\in\mathcal{I}}\sum_{u\in\mathcal{U}} c_{ij}^{(v)}x_{iju} \leq B_j, \quad j \in \mathcal{J} \tag{4.14}$$

Some alterations have been made to constraint set (4.3) defined in the baseline IP formulation model to derive constraint set (4.15). The purpose of constraint set (4.15) remains to limit the number of products j being offered to customers i. A new term $(1 - r_i^{(c)})$ is, however, introduced to constraint set (4.15), which enables the model to perform recency checks before allowing any product to be assigned to a customer. Recency is a compliance requirement that must be considered when running direct marketing campaigns. It refers to the time needed to pass from the previous direct marketing interaction before a financial institution can send follow-up direct marketing communications. However, the recency period varies with the communication channel being utilised. Note that $r_i^{(c)}$ is defined as a user input parameter that takes on binary values to guide the model in its recency decision process. When $r_i^{(c)}$ is assigned a value of 0, the left-hand side of constraint set (4.15) takes on a value of 1 and allows the optimisation model to allocate at maximum 1 product to customer i. If a value of 1 is assigned to $r_i^{(c)}$, the left-hand side calculation of constraint set (4.15) will equate to 0. An upper bound limitation of 0 for constraint set (4.15) would mean that for the customer under consideration, no product may be assigned to this customer as he or she would have already received an offer within the prescribed recency period. Only after a predefined period has lapsed will the optimisation model be allowed to offer customer i another product j. We do not track recency periods for the IP formulation but take the binary recency parameters as user input. Channel index u is added to constraint set (4.15) to account for channel decision complexities.

$$\sum_{j\in\mathcal{J}}\sum_{u\in\mathcal{U}} x_{iju} \leq (1 - r_i^{(c)}), \quad i \in \mathcal{I} \tag{4.15}$$

To limit the number of customers i assigned to each product j, we again introduce constraint sets (4.5) – (4.6) to the IP formulation model. The preceding constraints are, however, altered in order to account for channel decisioning by adding index u, which results in the derivation of constraint sets (4.16) – (4.17). Besides the index addition, no additional mathematical changes have been made to constraint sets (4.16) and (4.17). Similar to constraint set (4.5), constraint set (4.16) still represents the lower bound limit of customers being assigned to a specific product j. In contrast, constraint set (4.17) denotes the upper limits as was mentioned for constraint set (4.6).

$$\sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} x_{iju} \geq l_j^{(l)} y_j, \quad j \in \mathcal{J} \tag{4.16}$$

$$\sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} x_{iju} \leq l_j^{(u)} y_j, \quad j \in \mathcal{J} \tag{4.17}$$

Some parts of the novelty aspects of the proposed product targeting problem were mentioned in the above paragraphs. In addition to the already mentioned novelty proposals, we introduce constraint sets (4.18) – (4.29) as constraints, which will allow the optimisation model to account for many new business and operational constraints that try to align the model to real-world product targeting problems.

As an introduction to the constraints, consider constraint set (4.18). This constraint is added to the optimisation model to limit the number of products allowed to be considered during the decision process. Parameter $P^{(m)}$ represents a user input upper bound for constraint set (4.18), which denotes the limit imposed on the number of products that could be offered. Suppose there is a list of 10 products available to be committed by the optimisation model, for example. In that case, $P^{(m)}$ could be assigned a value of 5, which will force the optimisation model only to consider the best combination of 5 or fewer products and prevent the remaining products from forming part of the optimisation model decisioning framework. Including constraint set (4.18) into the optimisation model allows the financial institution to provide the model with all of the available products that could be offered. However, it allows for dynamically changing $P^{(m)}$ to increase or reduce the pool of products to choose from. If this constraint is not considered, the end user of the optimisation model would need to manually upload the exact number of products that would need to be considered for each model run, as the model would not have the capability of limiting product selection if the need arises. Given the structure of constraint set (4.18), it is not necessary to include the channel index u as product selection is independent of the communication channel.

$$\sum_{j \in \mathcal{J}} y_j \leq P^{(m)} \tag{4.18}$$

As part of channel selection, constraint set (4.19) is added to the optimisation model in order to allow only the selection of one channel u as means of communicating offer j to customer i. When performing channel selection, the model can select any of the following three channels, which include voice ($u = 1$), SMS ($u = 2$), or email ($u = 3$) as long as it adheres to the prescribed channel limitations and preferences specified by the end user. There are, however, multiple new and upcoming channels within financial institutions that one could also incorporate into the modeling framework, such as push notifications, in-app messaging, and USSD, to name a few. However, for this thesis, these channels and the complexity associated with each were not considered. Therefore, the right-hand side of constraint set (4.19) is assigned a value of 1 to ensure single channel selection. However, if the end user would like more than one channel to be assigned to customers for a specific offer, the right-hand side constant could be changed to a value greater than 1.

$$\sum_{u \in \mathcal{U}} x_{iju} \leq 1, \quad j \in \mathcal{J}, i \in \mathcal{I} \tag{4.19}$$

With constraint set (4.19) included, the model can assign a channel to each customer. However, given channel limitations, some customers might need to be excluded from specific channels and

shifted towards another channel. For example, a call center agent only has a limited number of customers i that he or she can contact in an 8-hour work day to sell a product j. Suppose the model assigns too many customers to voice as a channel, for example. In that case, the workload could be too much for the available agent to handle, resulting in leads being neglected or not attended to in the process. This will be detrimental to the monetary gain the financial company is pursuing. Another example could be that the number of SMSs sent to customers is limited to prevent spam or to reduce the cost of messaging. Similar to the mentioned examples, there exists a multitude of other reasons why one would like to limit the number of customers assigned to a specific channel. The preceding is accounted for by including the upper bound channel limiting constraint set (4.20). In constraint set (4.20), $c_u^{(m)}$ denotes the input parameter used to manage the upper bound limits.

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} x_{iju} \leq c_u^{(m)}, \quad u \in \mathcal{U} \tag{4.20}$$

Like the upper bound limitations, the model must govern the minimum number of leads assigned to each channel. A minimum channel limit is essential as some channels might have monthly sales quotas to be achieved. With no minimum lead requirement, the end user runs the risk of the model only assigning leads to the cheapest or most successful channels and eliminating/killing the other channels by distributing no leads to those channels. For example, if the end user only focused on cost and did not impose minimum limits, the optimisation model would send all of the leads to the SMS and email channels as they are much cheaper than the voice channel. From a business and execution contracting perspective, this would be unacceptable; therefore, a minimum lead requirement for each channel is imposed. Parameter $c_u^{(l)}$ in constraint set (4.21) denotes the lower bound, which is used to feed the lower limit data points into the optimisation model.

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} x_{iju} \geq c_u^{(l)}, \quad u \in \mathcal{U} \tag{4.21}$$

Additional constraints need to be added to the optimisation model to manage the number of products j being offered to the relevant customers via each channel. Note that constraint sets (4.20) – (4.21) only impose limits on the total events that can take place per channel but do not specify any limitations on specific events (i.e., limit the number of a specific product j being offered through channel u). For this reason, constraint sets (4.22) – (4.23) are introduced. Without these constraints, the model can satisfy constraint sets (4.20) – (4.21) by occupying all the available events of a specific channel with only one or two products. These will typically be products with a high-profit margin or low offering costs. As a financial institution, it would not make sense to promote only one or two products as the institution would want to increase the entrenchment of customers into the company. The more products customers have with the financial institution, the more likely they will use the available services and the less likely they will leave the organisation as a service provider. In order to enforce the upper bound limit for each product of a specific channel, a constraint set (4.22) is introduced. E_{ju} is used as an exclusion parameter, meaning that when a business unit requests that a specific product should not be offered via a particular channel to a customer, then the parameter will take on a value of 1. When $E_{ju} = 1$, the LHS term of constraint set (4.22) will equate to zero, which would prevent the model from selecting the product and channel under consideration. When E_{ju} is, however, allocated a value of 0, then the value of $l_{ju}^{(m)}$ is assigned as the upper bound limit which governs the number of each product j that could be offered via the different channels u.

$$\sum_{i\in\mathcal{I}} x_{iju} \leq (1 - E_{ju}) l_{ju}^{(m)}, \quad j \in \mathcal{J}, u \in \mathcal{U} \tag{4.22}$$

The lower bound limit for each product offered via a specific channel is managed by a constraint set (4.23). Note that input parameter E_{ju} has been added to this constraint as well in order to account for the exclusion of certain products from selected channels if required by a specific business unit. Parameter $l_{ju}^{(n)}$ represents the lower bound input parameter. When $E_{ju} = 0$, then $l_{ju}^{(n)}$ will guide the model in the decision process in order to meet the minimum business requirements. An additional decision variable (y_j) is added to constraint set (4.23). This product commitment decision variable

is added in order to only allow the offering of a product via a specific channel if the model has decided to include the specific product into the decision framework (i.e., $y_j = 1$). If y_j was not included in constraint set (4.23), the model could experience infeasibility conditions, for example, when the optimisation model is not considering a product. However, the constraint still requires a minimum number of this specific product to be offered via a certain channel. It will be impossible for the model to satisfy this condition, and the problem would be infeasible. For this reason, y_j is added to constraint set (4.23). By adding constraint sets (4.22) – (4.23) to the optimisation model, the product selection and offers via the various channels represent the entire product range for the financial institution and thus prevent single product offering biases. It also allows the model to satisfy stringent business requirements regarding product and channel decisions.

$$\sum_{i \in \mathcal{I}} x_{iju} \geq (1 - E_{ju}) l_{ju}^{(n)} y_j, \quad j \in \mathcal{J}, u \in \mathcal{U} \tag{4.23}$$

Constraint sets (4.22) – (4.23) were added to manage the exclusion of selected products from specific channels if the business requirement arises. However, the model also needs to be capable of excluding customers from certain channels if the customer has specified a preference toward a specific channel. For example, a customer might prefer being contacted via email instead of receiving an SMS or being called telephonically. In such an event, the model would need to prioritise the email channel as the primary contact medium, given that input has been received directly from the customer. Ignoring customer preference could result in poor client experience and increased churn rates. Therefore, constraint set (4.24) is introduced with $c_{iu}^{(e)}$ being a binary input parameter used to guide the mathematical model regarding the channel preference of customers. When $c_{iu}^{(e)}$ is assigned a value of 1, it will result in the right-hand side term of constraint set (4.24) to take on a value of 0. The preceding will prevent the model from selecting the channel under consideration for a given customer, as it has been specified that the customer does not prefer to be contacted via the said channel. In the event that $c_{iu}^{(e)}$ equates to a value of 0, the right hand side term will constitute only of m_3. Parameter m_3 has been defined as a very large constant in order to ensure that when a customer has not provided a specific channel exclusion ($c_{iu}^{(e)} = 0$), that the constraint will still hold and will not lead to model infeasibility. Let us say, for example, that m_3 was not included in constraint set (4.24). This would mean that if $c_{iu}^{(e)} = 0$, then the right-hand side of constraint set (4.24) would equate to 1. The preceding is fine if a customer is only allowed to be assigned 1 product offering j (refer to constraint set (4.15)). However, if constraint set (4.15) is updated to allow more than 1 product j to be assigned to customer i, then constraint set (4.24) would not allow this decision capability unless a large constant parameter m_3 is added to constraint set (4.24). If m_3 is not added to constraint set (4.24), then these two constraints will conflict with one another.

$$\sum_{j \in \mathcal{J}} x_{iju} \leq m_3 (1 - c_{iu}^{(e)}), \quad i \in \mathcal{I}, u \in \mathcal{U} \tag{4.24}$$

From a direct marketing perspective, the mathematical model also needs to take into consideration customer marketing consent when allocating products j via channels u to various customers i. Constraint sets (4.25) and (4.26) are incorporated to include marketing consent as a framework in the optimisation model. In constraint set (4.25), decision variable $c_{ijuq}^{(d)}$ is added to the left-hand side of the constraint while setting the mentioned variable less than or equal to (1 - $c_{ijuq}^{(pr)}$). As deliberated in Section 4.3.1, decision variable $c_{ijuq}^{(d)}$ keeps track of the best number or email address selection for a customer. If input parameter $c_{ijuq}^{(pr)}$ is assigned a value of 1, meaning that the customer indicated no to marketing for a given number or email address, the associated decision variable $c_{ijuq}^{(d)}$ will be forced to take on a value of 0. Alternatively, if $c_{ijuq}^{(pr)} = 0$ then $c_{ijuq}^{(d)}$ will not be forced to a value of 1, but rather be allowed to take on either value of 0 or 1 depending on the influence of constraint set (4.26). Note that there are different ways in which one could model marketing consent decisioning. One could either eliminate a customer if he or she has declined marketing consent, or one could alternatively discount the number or email on which the consent was reported.

The preceding is, however, dependent on the type of data available to the organisation implementing such an optimistion model and the legislative requirements the company must adhere to (for example, POPIA). For this thesis, marketing consent was modeled on a number or email address basis and only eliminated the channel on which marketing consent was rejected. In turn, the model was still allowed to offer product j to customer i while allowing the model to use an alternative channel where marketing consent was received (yes to marketing, $c_{ijuq}^{(pr)} = 0$).

$$c_{ijuq}^{(d)} \leq (1 - c_{ijuq}^{(pr)}), \quad j \in \mathcal{J}, i \in \mathcal{I}, u \in \mathcal{U}, q \in \mathcal{Q}_{iu} \tag{4.25}$$

As part of the marketing consent modeling strategy, constraint set (4.26) is added to ensure that the decisioning between variables $c_{ijuq}^{(d)}$ and x_{iju} are aligned. Note that variable $c_{ijuq}^{(d)}$ is only allowed to take on a value of 1 if variable x_{iju} has been assigned a value of 1. Meaning the optimisation model should first decide which product j should be offered to customer i through channel u, before decision variable $c_{ijuq}^{(d)}$ makes a call on the best number or email address q for the selected channel u. Without the limitations imposed by constraint set (4.26), decision variable $c_{ijuq}^{(d)}$ will be allowed to take on binary variables independent of the dynamics of the rest of the optimisation model. Therefore, constraint set (4.26) must be added to the model to keep the interactions between the various decision variables interconnected and dependent on each other. Constraint set (4.26) also ensures that when selecting the best number or email address for a customer, only 1 option is selected per channel. The foregoing is enforced by setting the summation of $c_{ijuq}^{(d)}$ per index Q_{iu} equal to x_{iju}. For example, if the model has selected voice as a channel for a customer and the customer under consideration has 3 cellphone numbers. Constraint set (4.26) will only allow $c_{ijuq}^{(d)}$ to select 1 of the 3 available numbers. It would not make sense for our problem to allow multiple numbers or email selections, so only a single channel selection is considered. Multi-selection will probably make sense to include if the business unit wants to obtain a list of numbers in descending order of importance. This example applies when a number is used to phone a customer, but the first attempt terminates in a cold call. In this instance, the call center agent would have alternative numbers to consider to get hold of the client. However, for the research scope, the preceding was separate from this study.

$$\sum_{q \in \mathcal{Q}_{iu}} c_{ijuq}^{(d)} = x_{iju}, \quad j \in \mathcal{J}, i \in \mathcal{I}, u \in \mathcal{U} \tag{4.26}$$

When including channel as part of the problem, the model is also required to decide on the best time of day to call a customer if voice ($u = 1$) has been selected as the communication medium of choice. Note that time of day does not really apply to SMS and email as a channel; therefore, constraint sets (4.27) – (4.28) are only limited to voice. However, time selection happens in two phases in the mathematical model, with the first type of time selection happening in constraint set (4.27). At this stage, the model determines if the customer was selected to receive an offer via voice. If true, the model is not concerned about the best time or number, but just that it would be required to perform time selection for the customer under consideration. The next step in the time selection process occurs in constraint set (4.28), where a refined selection happens. In constraint set (4.28), the model is guided to select the best time to call the customer, identified in constraint set (4.27). Note that time selection in constraint set (4.28) is only necessary for the number selected to call the customer. The remainder of the numbers associated with the customer's profile will not be assigned a time flag, as those numbers will not be utilised to contact the said customer.

In constraint set (4.27), variable h_{jiuqt} is used to track the decision regarding the time of day to call a customer with no specific focus on the cellphone number itself at this stage. In constraint set (4.27), h_{jiuqt} is set to less than or equal to the product $m_4 x_{iju}$. If decision variable x_{iju} is assigned a value of 0 where index $u = 1$, no time of day selection would be required, as a result of voice being excluded from the list of available communication channels. The opposite is also true when $x_{iju} = 1$ where index $u = 1$, then the model will be allowed to start time selection from the list of available phone numbers associated to the customer's profile, as voice has been selected as the desired communication medium. The constant m_4 is added to the right-hand term in order to

allow the model to assign a value of 1 to decision variable h_{jiuqt} where index $u = 1$ for each number q and period t available to the customer. Note that m_4 takes on a large constant value to allow feasibility. As discussed in the previous paragraph, constraint set (4.27) is just used to indicate that time selection is required for a customer. Thus we would allow the model to select all numbers and available periods as 1 as an initial start to constraint set (4.27). With the inclusion of constraint set (4.28), the model will have a comprehensive list of periods t for each available number q to select from in order to refine h_{jiuqt} for each customer i.

$$\sum_{q \in \mathcal{Q}_{iu}} \sum_{t \in \mathcal{T}} h_{jiuqt} \leq m_4 x_{iju}, \quad j \in \mathcal{J}, i \in \mathcal{I}, u = 1 \tag{4.27}$$

Note that in constraint set (4.28), variable h_{jiuqt} is set to less than or equal to $c^{(d)}_{ijuq}$. As mentioned in the above sections, $c^{(d)}_{ijuq}$ is a decision variable used to select the best phone number or email address for a given customer. For constraint set (4.28), $c^{(d)}_{ijuq}$ is used to indicate which phone number has been selected as the best number to use for the selected customer. With $c^{(d)}_{ijuq}$ taking on a value of 1 if the phone number has been selected as the best number for use. This will therefore mean that decision variable h_{jiuqt} will only be allowed to select the best time of day to call a customer for the number which has been selected by $c^{(d)}_{ijuq}$. The summation of h_{jiuqt} over index $t \in \mathcal{T}$ is to ensure that only 1 period is selected for the phone number under consideration. Note that for the rest of the numbers linked to the customer's profile, no time selection will take place as $c^{(d)}_{ijuq}$ will be assigned a value of 0 for the associated numbers. The preceding will prevent the model from further decisioning on the excluded numbers while only focusing on the number selected for use.

$$\sum_{t \in \mathcal{T}} h_{jiuqt} \leq c^{(d)}_{ijuq}, \quad j \in \mathcal{J}, i \in \mathcal{I}, u = 1, q \in \mathcal{Q}_{iu} \tag{4.28}$$

The next constraint set is the last constraint introduced into the IP formulation of the product targeting problem (4.29). This constraint is related to the cross-sell capability discussed in Section 4.3.1. Decision variable o_{ijc}, which is used to track cross-sell option selection in constraint set (4.29), is set equal to or less than $c^{(s)}_{iju} x_{iju}$. $c^{(s)}_{iju}$ in constraint set (4.29) is a binary input parameter that is used to indicate if a cross-sell option exists for a certain product that has been offered to customer i via channel u. Note that if a cross-sell option exists, then $c^{(s)}_{iju} = 1$ whereas if not then $c^{(s)}_{iju} = 0$. If $c^{(s)}_{iju} = 1$ and $x_{iju} = 1$, it would mean that for the product which is offered to the selected customer, there exist a cross-sell option as well, and it allows decision variable o_{ijc} to either select the cross-sell option as a secondary offer for the customer ($o_{ijc} = 1$) or exclude it from the customer's offering completely ($o_{ijc} = 0$). Generally, when a cross-sell opportunity is available, the model will add it to the product offering as it will increase the monetary gain that could be achieved. It will increase the stickiness of the customer within the organisation. Note that constraint set (4.29) is only relevant to where $c^{(s)}_{iju} = 1$ in order to calculate the value of o_{ijc}. Where $c^{(s)}_{iju} = 0$, the given constraint does not have any influence on the product offering proposed by the model.

$$\sum_{c \in \mathcal{C}_{ij}} o_{ijc} \leq \sum_{u \in \mathcal{U}; c^{(s)}_{iju} = 1} c^{(s)}_{iju} x_{iju}, \quad j \in \mathcal{J}, i \in \mathcal{I} \tag{4.29}$$

The proposed IP formulation product targeting problem is an NP-hard problem that is very difficult to solve. Solving complexity either arises from memory limitations or exponential time requirements. The preceding was noted throughout the literature for similar product-targeting formulations with less complexity. Adding all of the preceding complexities to an already complex product targeting problem will likely increase the difficulty of finding a solution to this problem.

4.3.3 Novel IP formulation contributions

A summarised list of all the contributions added to the basic product targeting problem (Section 4.2) is provided. These contributions are primarily focused on adding new business and operational

constraints and expanding the indices of the product targeting problem in such a manner as to allow for the inclusion of time, channel, and cross-sell complexities. These contributions allow the model to align more closely with real-world problems. The contributions include:

1. Adding customer recency considerations to the base model framework,
2. Incorporating channel complexity to the product targeting problem, taking into account voice, SMS, and email as the method of communication,
3. Considering the best time to call customers if the optimisation model has selected voice as a medium of communication (morning, afternoon, and evening),
4. Accounting for product limitations per customer and channel,
5. Introducing channel limitations to the model,
6. Allowing customer channel preference to dictate model selection,
7. Including ML model contactability output (right party contact rates and answer probabilities) to allow the model to choose the best possible number or email address in the event of no customer preference being provided,
8. Inclusion of marketing consent to allow the optimisation model to either permit or deny customers to form part of the decisioning process,
9. Taking cross-sell opportunities and dynamics for selected products into consideration,
10. Formulating a mathematical model in a manner that allows newly defined decision variables such as time, channel, and cross-sell options to operate interconnectedly with the already defined product and customer decision variables,
11. Increasing overall model complexity by aligning the mathematical model closer to real-life operational and business limitations and requirements,
12. Improving model relevance within the product targeting industry.

As part of this thesis, a column generation approach is introduced to try and reduce the complexity of the IP formulation product targeting problem while still considering all of the aspects elaborated on throughout Section 4.3. In Chapter 5, a detailed overview is provided of the theory regarding the Dantzig-Wolfe decomposition, the column generation algorithm, and various stabilisation techniques available to improve performance. Chapter 6 contains a detailed discussion of the column generation approach.

Chapter 5

Mathematical principles

This chapter presents the theory related to Dantzig-Wolfe decomposition and column generation. The reader is provided with insights into the mathematical modeling techniques used to formulate the column generation product targeting problem, which will be introduced in Chapter 6. In order to elaborate on the theory, a mathematical problem containing an objective function and a set of constraints are considered, see (5.1) (Gamst, 2010):

$$\begin{aligned} \max \quad & cx \\ s.t. \quad & Ax \geq b \\ & x \in S \end{aligned} \tag{5.1}$$

In the above formulation, x denotes the decision variables which need to be computed by the optimisation model, with the values of x in the set S. Parameter c is the cost vector within the objective function multiplied by the decision variable x. Parameter A refers to a coefficient matrix linked to constraint set while b is the constraint lower bound vector. If decision variables x are allowed to take on only integer or binary values within the set S, the optimisation problem is referred to as an integer programming problem (IP). When x takes on real number values, the model would be known as a linear programming problem (LP). If, however, x is allowed to take on mixed values of both integer and real numbers, the associated model is a mixed integer programming problem (MIP). To explain the theory related to the Dantzig-Wolfe decomposition and column generation algorithms, we take $X \in R : x \geq 0, x \in \mathcal{X}$ so that the problem is transformed to an LP problem, with $\mathcal{X}$ referring to the collection of x. Formulation (5.1) is known as the primary optimisation problem. The theory related to the dual of the primal problem is also of importance for deriving the product targeting column generation algorithm. Problem formulation (5.2) is representative of the transposition of the primal problem (5.1) into its dual formulation (Gamst, 2010).

$$\begin{aligned} \min \quad & b^T y \\ s.t. \quad & A^T y \leq c^T \\ & y \geq 0 \end{aligned} \tag{5.2}$$

When depicting the parameters provided in (5.2), it should be noted that c^T, b^T and A^T refers to the transposed forms of c, b and A respectively. One crucial aspect to consider when solving both the primal and the dual optimisation formulations is that both problem instances will terminate at optimal solutions having the same objective function values. Now that the reader is familiar with both the primal and dual solution formulations, a detailed introduction to the theory relating to Dantzig-Wolfe decomposition and column generation can be presented.

5.1 Dantzig-Wolfe decomposition theory

When applying Dantzig-Wolfe decomposition, an original optimisation problem is divided into a master problem and associated pricing problems. The master problem typically consists of a reduced number of constraints compared to the original optimisation problem, which consists of an

increased number of columns. The purpose of the pricing problem is to generate new and improved columns capable of enhancing the solution quality of the master problem.

The problem under consideration must be transformable into the correct block angular structure to apply this decomposition methodology. An example of such a block-angular structure is provided in Figure 5.1. This figure indicates the optimisation problem containing linking/connecting and complicating/independent constraints. Linking constraints will constitute a summation across some or all of the available indices for a given decision variable or even for multiple decision variables which might be present within the mathematical model. Complicating constraints generally complicate the problem, whereas the problem's complexity is reduced significantly once removed. The complicating constraints can also be split into multiple smaller optimisation problems, as the influence of one constraint would not depend on the influence of another.

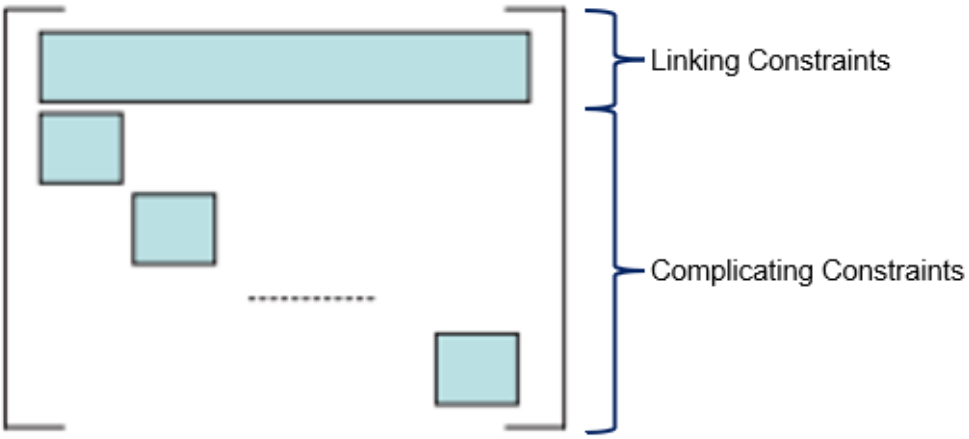

Figure 5.1: Block-angular Form, adapted from (Zhao, 2005)

The formulation of the block-angular structure presented in Figure 5.1 is provided in (5.3) where R refers to the set of blocks encompassed within constraint sets A^r and D^r respectively. Constraint set A^r is known as the linking constraints, whereas D^r refers to the complicating constraints (Gamst, 2010).

$$\begin{aligned} \max \quad & \sum_{r \in \mathcal{R}} c^r x^r \\ s.t. \quad & \sum_{r \in \mathcal{R}} A^r x^r \leq b \\ & D^r x^r \leq d^r, \quad r \in \mathcal{R} \\ & x^r \in \{0,1\}, \quad r \in \mathcal{R} \end{aligned} \tag{5.3}$$

The extended mathematical form of (5.3) is represented by (5.4). In this formulation, one can identify the difference between the linking constraints containing a summated set of decision variables. In contrast, the complicating constraints only consist of selected variables linked to each associated constraint (Zhao, 2005).

$$\begin{aligned} \max \quad & c^1 x^1 + c^2 x^2 + ... + c^r x^r \\ s.t. \quad & A^1 x^1 + A^2 x^2 + ... + A^r x^r \leq b \\ & D^1 x^1 \leq d^1 \\ & ... \qquad D^2 x^2 \leq d^2 \\ & ... \qquad\quad D^r x^r \leq d^r \\ & x^r \in [0,1], \quad r \in \mathcal{R} \end{aligned} \tag{5.4}$$

Once the LP problem is structured in the required block-angular structure, the problem can be decomposed into the relevant master and pricing problems. In (5.5), the master problem linked to

the primal optimisation problem is presented. Note that the master problem only consists of the objective function and the relevant linking constraints. The complicating constraints were removed from the master problem formulation. The complicating constraints will make up the various pricing optimisation problems, with the pricing formulation only being elaborated in Section 5.2. The master problem is structured in a manner that requires variables x^r to adhere to the complicating constraints left out from the problem formulation.

$$\begin{aligned} \max \quad & \sum_{r\in\mathcal{R}} c^r x^r \\ s.t. \quad & \sum_{r\in\mathcal{R}} A^r x^r \leq b \\ x^r \in X^r, \quad & r \in \mathcal{R} \quad where \quad X^r = [x^r \in [0,1], D^r x^r \leq d^r] \end{aligned} \tag{5.5}$$

The master problem is rewritten into a convex combination of its associated extreme points and rays, in (5.6), according to the Minkowski theorem. In this theorem decision variable x^r is rewritten using the extreme points x^i multiplied with a real number decision variable λ_i as well as the extreme rays y^j influenced by decision variable μ_j. Using the Minkowski theorem, it is required that the summation of λ_i equates to a value of 1 while both λ_i and μ_j takes on real numbers greater than zero (Lubbecke, 2010).

$$x^r = [x = \sum_i \lambda_i x^i + \sum_j \mu_j y^j | \sum_i \lambda_i = 1, \lambda_i \geq 0, \mu_j \geq 0] \tag{5.6}$$

When considering X^r as a bounded polyhedron, we can express X^r as a simplified version of (5.6) by excluding the extreme rays allowing the derivation of (5.7) where x^{rt}, t = 1,..., T_r are extreme points of X^r, with T_r referring to the collection of extreme points.

$$x^r = [x^r = \sum_{t\in\mathcal{T}_r} \lambda_{rt} x^{rt} | \sum_{t\in\mathcal{T}_r} \lambda_{rt} = 1, \lambda_{rt} \geq 0] \tag{5.7}$$

After rewriting x^r in its extreme point form, (5.6) can be substituted into (5.5) in order to transform the overall LP optimisation model into convex combinations of its extreme points. Refer to (5.8) – (5.11) for the Dantzig-Wolfe decomposed formulation of the master problem. Note that in this formulation, x^{rt} is transformed from a decision variable into a constant input parameter with λ_{rt} being utilised as the real number decision variable within the decomposed optimisation model. Constraint sets (5.10) and (5.11) are added to the master problem as part of the Minkowski theorem requirements (Pal, 2006).

$$\max \quad \sum_{r\in\mathcal{R}} \sum_{t\in\mathcal{T}} (c^r x^{rt}) \lambda_{rt} \tag{5.8}$$

$$s.t. \quad \sum_{r\in\mathcal{R}} \sum_{t\in\mathcal{T}} (A^r x^{rt}) \lambda_{rt} \leq b \tag{5.9}$$

$$\sum_{t\in\mathcal{T}} \lambda_{rt} = 1, \quad r = 1, ..., R \tag{5.10}$$

$$\lambda_{rt} \geq 0, \quad r \in \mathcal{R} \tag{5.11}$$

It is, however, imperative to note that in order to ensure that the master problem formulation is compatible with the column generation formulation as proposed in Chapter 6, constraint set (5.10) was augmented, as noted in (5.12). The preceding was done to ensure the applicability of the preceding formulation within the product targeting domain.

$$\sum_{t\in\mathcal{T}} \lambda_{rt} \leq 1, \quad r = 1, ..., R \tag{5.12}$$

Applying the augmentation seen in (5.12) to (5.10) allowed the model to eliminate specific product categories instead of committing all products for offering to the interested customer base. The preceding, in turn, ensures that the model is much more dynamic. The user can feed in all the available

product data. In contrast, the user leaves it up to the model to decide which products to commit to, given all the operational, business, and channel requirements. If the preceding is not included, the user would have been required to eliminate certain products prior to initialising and running the optimisation model, as the model would want to select an extreme point for each of the product offerings available in the model input.

The intricacies of the Dantzig-Wolfe decomposition algorithm are depicted in Figure 5.2. The first step is to identify the LP optimisation problem. Once the LP problem has been identified, it should be rewritten into a block-angular structure to ensure that the problem instance is compatible with the Dantzig-Wolfe decomposition algorithm. In the block-angular form, the problem is split into a master problem and multiple sub-problems. Then, the master problem is transformed using the Minkowski theorem to rewrite it as a convex combination of its extreme points. Finally, the master problem constraints are altered to fit into the product targeting structure as this thesis requires.

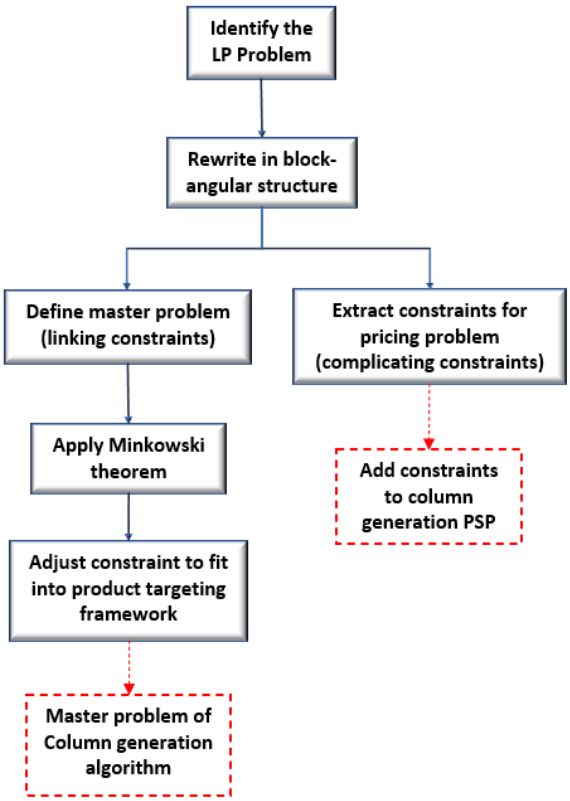

Figure 5.2: Dantzig-Wolfe algorithm

After generating the Dantzig-Wolfe decomposed formulation of the optimisation problem, it is apparent that there exist a significant number of columns. Therefore, by attempting to solve the given formulation, one would run into memory limitations due to the immense number of columns in the original formulation. To address this issue, a column generation algorithm is considered.

5.2 Column generation theory

The idea of a column generation algorithm is only to consider a subset of the columns in the master problem, seen in Section 5.1, instead of the entire search space of extreme points. This master problem is defined as the restricted master problem. Given that a profusion of columns would either take on zero values within the optimal solution or negatively influence the solution outcome, only a subset of columns should be considered. There will exist only a specific combination of columns that will have the potential to improve the current solution. The process of identifying the desired columns which should enter the basis of the master problem is known as column generation.

In order to explain the principles linked to the column generation algorithm, a restricted master problem (5.13) is defined (Gamst, 2010).

$$\begin{aligned} \max \quad & \sum_{j \in \mathcal{J}} c_j x_j \\ s.t. \quad & \sum_{j \in \mathcal{J}} a_j x_j = b \\ & x_j \in X \end{aligned} \tag{5.13}$$

In order to identify the columns which are required to enter the restricted master problem, the reduced cost associated with the restricted master problem is derived. The reduced cost associated with the restricted master problem defined in (5.13) is derived as seen in (5.14). Note that y constitutes the dual vector associated with the problem defined in (5.13) whereas c_j represents the objective function cost coefficients and a_j being indicative of the constraint coefficients (Gamst, 2010).

$$(c^j - y a_j) \tag{5.14}$$

The reduced cost derived in (5.14) is combined with the sub-problem independent constraints discussed in Section 5.1 to derive the complete pricing sub-problem. Finally, the comprehensive pricing problem formulation is provided in (5.15). Note that the reduced cost is encapsulated within a minimisation term to derive the objective function for the pricing problem. When the master or restricted master problem is defined as a maximisation problem, the pricing problem will form a minimisation problem formulation, as seen in (5.15). The opposite also holds when the restricted master problem is defined as a minimisation problem (Lubbecke, 2010).

$$\begin{aligned} \min \quad & (c_j - y a_j) \\ s.t. \quad & D x_j \leq b \\ & x_j \in [0, 1] \end{aligned} \tag{5.15}$$

When considering a minimisation pricing problem, a generated column will be able to improve the solution quality if a negative reduced cost is computed. In contrast, the opposite is valid for a maximisation problem. If the objective function value computed for a minimisation pricing problem is positive, it will indicate that the column is unable to improve the solution. The preceding algorithm can be presented in six possible steps as provided below (Pal, 2006):

1. Start and construct the initial restricted master problem.
2. Solve the restricted master problem.
3. Compute the dual values associated with each constraint contained within the restricted master problem to construct the reduced cost function, which will be used as the objective function for the pricing sub-problems.
4. Compute the sub-problem objective function values. If the obtained result is negative, it would be indicative of the existence of a column that is capable of improving the current solution.
5. If such a column exists, add the column to the reduced master problem and return to Step 2.
6. If however, no negative column is present in the solution, then the computational process should be terminated as the optimal combination of columns would have been identified.

The subsequent flowchart outlines the above steps within the column generation algorithm:

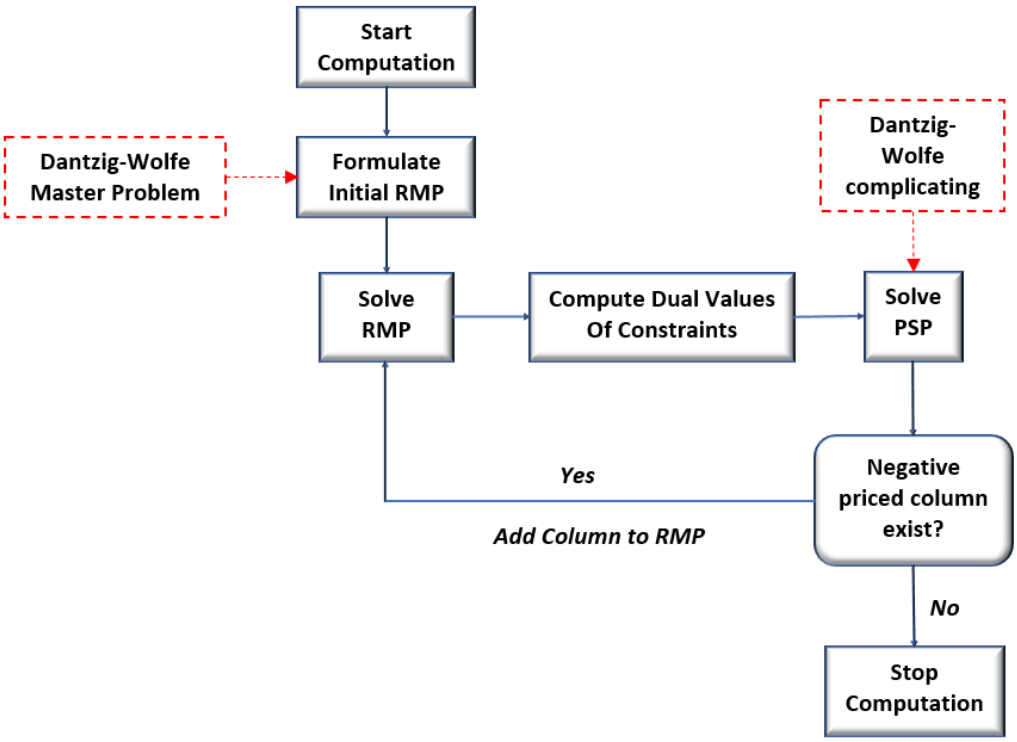

Figure 5.3: Column generation algorithm

When solving the reduced master problem (RMP), it is essential to note that the solution algorithm which should be utilised is known as the Simplex method. This method solves LP optimisation problems to optimality while enabling the extraction of dual values for each constraint contained within the RMP. The dual values are required to formulate the reduced cost objective function of the pricing sub-problem (PSP). Given that the PSP are groupings of IP problems, the branch-and-bound algorithm can be used to solve each of the sub-problems. Once the PSP problems are solved, the algorithm enters the loop structure, as depicted in Figure 5.3, until its termination conditions are satisfied.

It is imperative to note that both the Dantzig-Wolfe decomposition and column generation algorithms were initially independently formulated from one another in the literature. However, leveraging each of the algorithms' strengths allows complex optimisation problems to be solved more efficiently. It is noted, however, that the column generation algorithm struggles with behaviors known as heading-in, yo-yo, and tailing-off effects. The preceding may result in undesired performance deterioration of the column generation algorithm. Stabilisation techniques are proposed in the literature to counteract these effects.

5.3 Stabilisation techniques

When considering the performance of the standard column generation algorithm presented in Section 5.2, it was noted that this algorithm has some limitations, which results in the algorithm being very slow in computing the global optimal solution for a problem. Some of the issues identified include aspects such as seen in the subsequent figure:

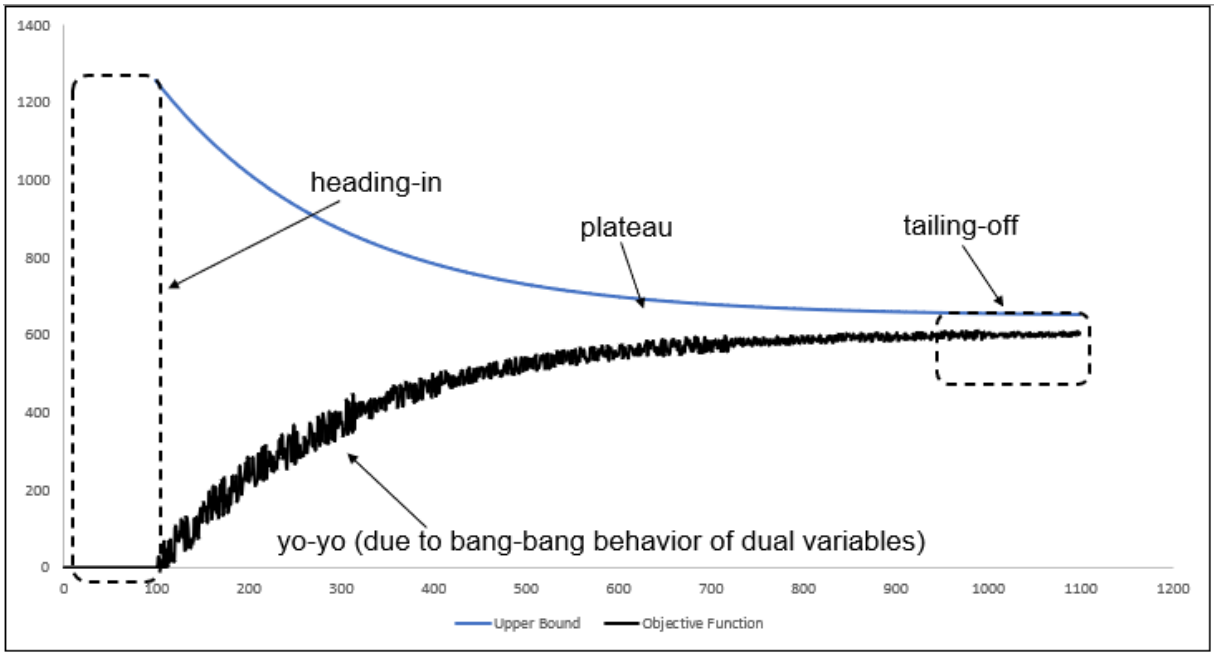

Figure 5.4: Column Generation Limitations (Vanderbeck, 2005)

In Figure 5.4, the various phenomena responsible for the deterioration of the standard column generation algorithm are illustrated with respect to bound calculations. Details regarding these phenomena are provided below (Vanderbeck, 2005; Pessoa *et al.*, 2017):

1. The first phenomenon is known as the heading-in effect, where the initial columns present in the algorithm are very poor. This prevents the column generation algorithm from generating new columns at the start, which could improve the current optimal solution. It does, however, delay the computational process, leaving the algorithm to run for many iterations before reducing the solution gap toward global optimality.

2. After exiting the heading-in effect, the column generation algorithm may enter a yo-yo or bang-bang state where the master problem objective function values fluctuate drastically. This can be attributed to the algorithm jumping from one extreme point to another to find the combination of extreme points that will provide the global optimal solution. As a result of the yo-yo/ bang-bang effect, the algorithm is exposed to unnecessary iterations as the algorithm tries to progress through the volatile dual values.

3. The column generation algorithm may also suffer from degeneracy, resulting in the reduced master problem objective function value remaining the same across multiple iterations. The algorithm enters this phenomenon right after the yo-yo effect, just before entering the tailing-off effect.

4. As the column generation algorithm converges to the global optimal solution, the conversion rate may slow down. The preceding is known as the tailing-off effect. At this point of the computational process, the algorithm must complete a significant number of iterations to induce a marginal change in the master problem objective function value.

Each of the phenomena identified in the above discussion contributes to the decline in the computational performance of column generation. Furthermore, the various aspects have a compounding effect resulting in an exponential increase in the compute time required by the column generation algorithm to find a suitable global optimal solution. Throughout the literature, various methodologies are proposed to manage the shortcomings of the standard column generation algorithm. Some of these techniques include interior point stabilisation, the boxstep method, polyhedral penalty terms, bundle methods using quadratic penalty terms, convex combinations with previous dual solutions,

valid inequalities in the dual space, and dual smoothing techniques. In this thesis, an altered version of the dual smoothing methodology is proposed to stabilise the column generation algorithm reported in Chapter 6.

5.4 Dual smoothing

The dual smoothing technique proposed by Neame (2000) is based on "correcting" the current dual solution (π^t) with that of the previous dual solutions ($\tilde{\pi}^{t-1}$) generated by the column generation algorithm when solving the master problem. The dual update rule suggested by Pessoa *et al.* (2017) is,

$$\tilde{\pi}^t = \alpha\tilde{\pi}^{t-1} + (1-\alpha)\pi^t \tag{5.16}$$

The above equation eludes to the fact that $\tilde{\pi}^t$ is a value that is representative of the sum of previous iterates where $\tilde{\pi}^t = \sum_{j\in\mathcal{J}}(1-\alpha)\alpha^{t-T}\pi_j^T$. At the same time, discounted factors are applied using α to model older candidates' obsolescence. Parameter $\alpha \in [0,1)$ influences the extent to which the dual values are being smoothed. After applying the dual smoothing technique to the various dual values, the pricing optimisation problem is solved using $\tilde{\pi}^t$ as part of the objective function instead of π^t. Refer to the new pricing problem objective function value (5.17) (Pessoa *et al.*, 2017).

$$z_{\tilde{\pi}^t} = argmin_{x\in Z}((c-\tilde{\pi}^t A)x) \tag{5.17}$$

In applying the dual smoothing methodology to the pricing sub-problems, the sporadic changes in dual values experienced by the column generation algorithm are reduced and, as a result, assist in stabilising and speeding up the column generation computations. Section 6.3.7 presents the dual averaging technique implemented in the column generation algorithm. The theory relating to Dantzig-Wolfe decomposition, column generation, and dual smoothing techniques, as discussed in this chapter, will be practically applied in Chapter 6 to derive the column generation algorithm, which will be utilised within the product targeting domain.

Chapter 6

Novel column generation approach

6.1 Introduction

The mathematical indices discussed in Section 4.1 are relevant for deriving the column generation master and sub-problems. An additional index set $\mathcal{W}$ is introduced, representing the columns added to the master problem.

In the subsequent section, the concept of parallel processing is introduced. By implementing this computational technique into the column generation algorithm, compute time required to obtain a global optimal solution to the product targeting problem is significantly reduced. The remainder of the chapter is devoted to the proposed column generation framework.

6.2 Parallel processing

Parallel processing refers to the process or methodology of separating large complex tasks into smaller tasks, which are executed simultaneously using multiple central processing units (CPUs). A practical example of parallel processing is depicted in the below figures.

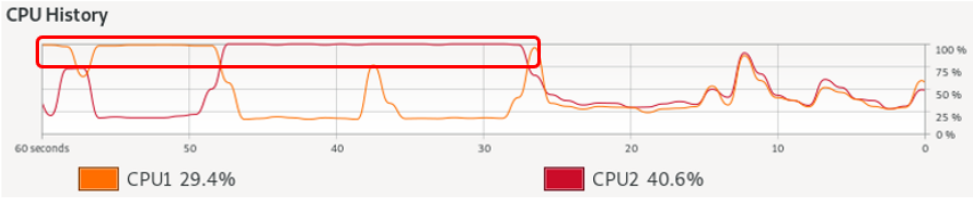

Figure 6.1: Single Processing Example

In Figure 6.1, two CPUs are available to perform tasks, namely CPU1 and CPU2. In this example, we do, however, see that only one CPU is committed to operation (100% operating capacity) at any given point in time. This is an example of single processing where only one of the two CPUs is utilised when intensive computations are executed. Generally, a computer will operate in this manner unless a parallel processing programming methodology is followed.

The example in Figure 6.2 demonstrates the capability of parallel processing, which allows the computer to simultaneously commit both CPUs to full operating capacity. The preceding emphasizes that when structuring code in such a manner to enable parallel processing, the computer will be capable of using multiple cores, if available, to speed up the time required to process intensive computations.

Figure 6.2: Parallel Processing Example

A detailed explanation of the mathematical formulation methodology followed to include parallel processing to the product-targeting column generation solution algorithm will be discussed in detail throughout Section 6.3.4. In Section 6.3, we first focus on the intricacies of the proposed column generation algorithm.

6.3 Column generation solution framework

The theory behind the product targeting column generation algorithm has been discussed in detail throughout Chapter 5. This section applies the Dantzig-Wolfe decomposition and column generation theories to the product targeting problem. The aim of utilising these theories is to reduce the complexity of the problem and obtain an optimal solution for problem sizes where the IP formulation, presented in Section 4.3, fails to do so.

In order to make use of column generation to solve the product targeting problem as defined in Section 4.3, the integer programming problem is transformed into a linear programming problem. To achieve this, we employ the Dantzig-Wolfe decomposition algorithm to transform the product targeting problem. Unfortunately, many variables are generated in the mathematical model obtained after applying the Dantzig-Wolfe decomposition transformation. Therefore, the only way to effectively solve the reformulated optimisation problem is to use the column generation algorithm.

As part of implementing the column generation algorithm, the reformulated Dantzig-Wolfe decomposed product targeting problem is divided into a master and multiple sub-problems. To allow the model only to consider a subset of the extreme points and rays at the start of the solution algorithm and significantly reduce the problem's computational complexity. The algorithm requires a starting solution to initiate the column generation method. The starting solution is generated using a specifically designed heuristic algorithm. The algorithm used to solve the master problem, a linear programming problem, is known as the simplex method. After solving the master problem using the simplex method, the duality and reduced cost theorems, discussed in Chapter 5, are used to formulate the sub-problems' objective functions. The sub-problems are well-defined integer programming problems solved using standard optimisation algorithms such as the branch-and-bound method. By solving the sub-problems, the algorithm generates new columns capable of improving the master problem's objective function (i.e., computation of more suitable columns closer to the global optimality point). The preceding columns are added to the master problem iteratively. When a column does not improve the objective function of the given problem instance, it is not added to the master problem. The algorithm then enters a computational process to select the best possible combination of columns from the list of columns in the master problem. By only adding and selecting the columns capable of improving the master problem's objective function, the optimisation model's size is significantly reduced compared to the IP formulation discussed in Section 4.3. This, in turn, improves the computational efficiency of the algorithm and reduces the model's memory requirements. If no new columns can be added to the master problem to improve the objective function, the process is terminated, and the optimality gap is calculated. As per the simplex algorithm, when no new columns can be added to the master problem, the algorithm has obtained the global optimal solution.

The focus is initially on providing extensive detail regarding the heuristic algorithm developed to feed the master problem with a starting solution to understand the complexities associated with

the proposed product targeting column generation algorithm. After that, information is provided on the primal master problem, its dual and reduced cost derivations, and the formulation of the sub-problems related to the master product targeting problem. We also provide some insights into the upper bound calculation for the master problem, and the various gap calculations used to determine how far the obtained solution is from global optimality. Finally, the mathematical formulation and derivation for each phase within the column generation algorithm are provided throughout the subsequent sections.

6.3.1 Model starting heuristic

The starting solution, which is required to initiate the column generation computational process, forms an integral part of the overall product targeting solution framework. The reason is that if one does not have an algorithm capable of providing an excellent initial feasible solution to the complex product targeting problem, the column generation algorithm on its own is rendered useless as an initial solution is required to initiate the computational process. For this reason, multiple avenues were investigated to find a suitable starting solution capable of delivering a good initial feasible solution. The initial starting heuristic was to use an altered version of the IP formulation product targeting formulation, as noted in Section 4.3. The starting heuristic consisted of the entire set of constraints seen in Section 4.3. However, the objective function was replaced with a value of 0. When solving the preceding formulation, the mathematical model would not try to optimise the objective function as it would be fixed to a value of 0. However, it would instead provide an output for the various decision variables that adhere to the mathematical model's constraints. The solution obtained from this initial starting heuristic is a feasible solution to the complex product targeting problem. The solution obtained will, however, not be very good as no optimised decision took place in the generation of the solution. The algorithm only focused on generating a solution that met the model requirements. Another limitation to this initial proposed solution was that to generate an initial feasible solution using this methodology, the model was still required to consider all the permutations and combinations contained within the full-scale product targeting problem. Considering all of the variables within the said problem, the starting heuristic ran into memory limitations when this methodology was applied to large-scale product targeting problems. For this reason, the preceding heuristic was only utilised for initial small-scale testing of the column generation algorithm to ensure that the results obtained from the overall solution framework were comparable to the baseline IP and IP product targeting formulations (i.e., verification testing).

A greedy starting algorithm was considered to improve the initial heuristic solution framework. In Section 6.3.1, a detailed pseudocode formulation is derived for the proposed starting heuristic, which was utilised in this thesis to generate an acceptable starting solution for the column generation framework. The proposed heuristic consists of the main function, which is used to manage the overall execution logic of the solution framework as well as sub-functions which is responsible for the actual inner workings and decision variable selection process of the algorithm. Line 1 of the starting heuristic is used to initialise variable $x^{(f)}_{piju}$. This variable is used to track the heuristic algorithm selection and updates for the decision variable x_{iju} as it moves through the various stages of computation. Line 2 of the pseudocode represents the initialisation of various decision variables $c^{(d)}_{ijuq}$, h_{jiuqt} and o_{ijc} as well as variables used to track matrix indexes such as $p^{(c)}$, f^f and f^P. Note that in this starting heuristic algorithm, we do not compute values for decision variables $c^{(d)}_{ijuq}$, h_{jiuqt} and o_{ijc}. The preceding decision variables are assigned zero values as a starting point for the column generation algorithm. In this starting heuristic, we only focus on calculating starting values for decision variables x_{iju} and y_j as these variables are critical to the initialisation of the column generation algorithm. Index variables $p^{(c)}$ and f^f are used to initialise the size of array $funct[p^c][f^f]$ in line 3 of the mathematical algorithm. Array $funct[p^c][f^f]$ contains the various number of permutations for constraint sets (4.15), (4.17), (4.21), (4.22), and (4.24). Note that when considering five constraints in this heuristic algorithm, the value of p^c is equated to a value of 120, which is representative of the number of permutations in which the mentioned constraints can be structured. The value of f^f

equates to 5, which denotes the number of constraints which is considered in this heuristic. If the number of constraints considered in the first phase of the heuristic increases, the values of p^c and f^f will need to be updated accordingly. The greedy heuristic algorithm is required to run through all of the various constraint permutations in order to try and determine the best selection of x_{iju} and y_j, which will lead to the highest possible objective function value for the product targeting problem. Similar to array $funct[p^c][f^f]$, we also define a secondary array $funct1[f^p]$ which is used to contain the list of constraint sets (4.19) and (4.14). Note that the algorithm first utilises array $funct[p^c][f^f]$ to perform variable selection on decision variables x_{iju} and y_j where after these variables are fine-tuned using constraint array $funct1[f^p]$. To track the change in the objective function as the algorithm progresses through its computational journey, we initialise variable $obj^{(new)}$, which is used to store the value of the objective function at each step of the iterative process. The value contained within $obj^{(new)}$ will be used throughout the algorithm to decide whether decision variables x_{iju} and y_j should be updated with the latest combination of binary variables or not.

Lines 5 to 13 in the pseudocode formulation points to the initial series of ranking and selection functions variable $x^{(f)}_{piju}$ is exposed to in order to determine a good combination of binary variables that would lead to a feasible solution for the product targeting problem. The heuristic algorithm starts with a for loop of the number of permutations defined in line 2. When entering the main for loop of the algorithm, there exist two subsets of for loops (lines 8–10 and 11–13). The first for loop (lines 8–10) is an iteration from 1 to 5 as per the number of constraints considered in array $funct[p^c][f^f]$. The order in which the 5 constraints are considered for the iteration under consideration depends on the permutation number p obtained from the main for loop. In the first for loop, the sub-function $F(p, x^{(f)}_{piju}, UB)$ is executed for each iteration of the loop. The sub function $F(p, x^{(f)}_{piju}, UB)$ is responsible for the ranking and selection of $x^{(f)}_{piju}$ while adhering to the limitations imposed by each constraint combination contained in $funct[p^c][f^f]$. More detail regarding the sub-function algorithm will be provided in the below section. After exiting the first for loop, the model has obtained an initial estimation of $x^{(f)}_{piju}$. Note that the binary variable combination contained in $x^{(f)}_{piju}$ at this stage is, however, only an initial stab at determining the best values for x_{iju}. In order to refine the selection of $x^{(f)}_{piju}$, the algorithm enters into the second for loop (11–13). In this for loop, the model again calls the sub-function $F(p, x^{(f)}_{piju}, UB)$ but at this stage, only constraint sets (4.19) and (4.15) are considered. Note that the algorithm does not run through different permutations of these two constraints and only ensures that the values selected in $x^{(f)}_{piju}$ still adhere to constraint sets (4.19) and (4.15). If the values contained in $x^{(f)}_{piju}$ violate the limitations imposed by either constraint set (4.19) or (4.15), the values for $x^{(f)}_{piju}$ are altered by the sub-function $F(p, x^{(f)}_{piju}, UB)$ as to ensure that the column generation algorithm receives an initial feasible solution to the product targeting problem at all times.

At this point in the heuristic algorithm, it is important to note that the constraints considered are only a subset of those defined in Section 4.3. The reason for this will become apparent when discussing the primal master problem formulation stipulated in Section 6.3.2. At this stage, it is only important to understand that for the heuristic formulation, we only need to consider the constraints that will form part of the column generation master problem. The reason is that the solution obtained from the heuristic algorithm will serve as the initial feasible starting solution to the master problem within the column generation formulation. Suppose the proposed solution violates the constraints within the column generation sub-problems at this stage. In that case, it will not pose a problem as those proposed combinations will be excluded from the selection process as we progress through the column generation solution framework. One could consider the entire constraint set within the heuristic framework. However, it will increase the complexity and computational requirements of the algorithm extensively while not necessarily providing a significant improvement to the proposed initial feasible solution. After executing the logic presented in lines 8 – 13, the algorithm moves to lines 14 – 17, which is implemented to ensure that the heuristic adheres to the corporate

hurdle constraint set (4.13) as discussed in Section 4.3. Line 15 is used to initialise variable *cost*. This variable is utilised in line 16 of the algorithm to compute the cumulative fixed cost associated with offering products j via the product targeting problem. The line 17 of the heuristic algorithm portrays the core calculation associated with the corporate hurdle constraint. The algorithm takes the difference between the campaign profit and expenses for each index i, j, u to ensure that the delta equals a positive value. If the expenses are more than the profit, the associated $x_{piju}^{(f)}$ value is assigned a value of 0 to prevent the corporate hurdle constraint from being violated. If no constraint violation is noted for constraint set (4.13), then $x_{piju}^{(f)}$ remains unchanged.

The next step in the heuristic algorithm is to identify values for the decision variable $y_j^{(heu)}$. During this process the values assigned to decision variable $x_{piju}^{(f)}$ is maintained constant even though the outcome of $x_{piju}^{(f)}$ is dependent on $y_j^{(heu)}$. Decision variable $x_{piju}^{(f)}$ will only be updated after $y_j^{(heu)}$ has been calculated. To compute $y_j^{(heu)}$ we introduce a simple optimisation problem formulation which is solved using the branch-and-bound algorithm. The optimisation formulation uses constraint set (4.9) as the maximisation objective function, a subsection of the detailed objective function discussed in Section 4.3.1. The problem is constrained using constraint set (4.18) to ensure that when assigning binary values to $y_j^{(heu)}$, the upper bound product limitations are taken into consideration. After solving the optimisation problem by implementing lines 19 to 21 in the pseudocode, a set of binary values is assigned to $y_j^{(heu)}$. After obtaining the results for $y_j^{(heu)}$, decision variable $x_{piju}^{(f)}$ needs to be augmented in order to account for potential product exclusions in the overall product targeting problem. For example, if in the optimisation problem $y_j^{(heu)}$ is assigned a value of 0 where $j = 2$, it would mean that variable $x_{piju}^{(f)}$ would not be allowed to take on a value of 1 when index j is equal to 2. Therefore all values of $x_{piju}^{(f)}$ where $j = 2$ would need to equate to 0 as per the logic presented in line 22 of the heuristic algorithm in order to prevent infeasible solutions from being provided to the column generation algorithm.

Algorithm 1: Heuristic Main Function Pseudocode

1: Let $x_{piju}^{(f)} = 1$
2: Let $c_{ijuq}^{(d)} = 0$; $h_{jiuqt} = 0$; $o_{ijc} = 0$; $p^c = 120$; $obj^{(new)} = 0$; $f^f = 5$; $f^p = 2$
3: $funct[p^c][f^f]$ = Compute 2D Matrix Containing Number of Permutations for constraints: 4.15; 4.17; 4.21; 4.22; 4.24
4: $funct_1[f^p]$ = Create 1D Matrix of Constraints 4.19 and 4.14
5: **Initialise Heuristic Function:**
6: Send variable $x_{piju}^{(f)}$ through a series of ranking and selection functions
7: **for** $p \in p^c$ **do**
8: **for** $f \in f^f$ **do**
9: $x_{piju}^{(f)}$ = Call function $F(p, x_{piju}^{(f)}$,UB = LH of $funct[p][f])$
10: **end for**
11: **for** $a \in f^p$ **do**
12: $x_{piju}^{(f)}$ = Call function $F(p, x_{piju}^{(f)}$,UB = LH of $funct1[a])$
13: **end for**
14: **Ensure Corporate Hurdle is Adhered To, [4.13]:**
15: let $cost = 0$
16: $cost+ = c_j^{(f)}, \quad j \in \mathcal{J}$
17: **if** $(((p_{iju} - (c_{iju}^{(v)} + cost)R)x_{piju}^{(f)}) \leq 0)$ **then** $(x_{piju}^{(f)} = 0)$ **end if**, $\quad i \in \mathcal{I}, j \in \mathcal{J}, u \in \mathcal{U}$
18: **Determine Value for Decision Variable** $y^{(heu)}$ **Using simplistic Cplex Model:**
19: Maximise $(\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{u\in\mathcal{U}}((p_{iju} - c_{iju}^{(v)})x_{piju}^{(f)})(r_{iju} + c_{iju}^{(p)} + c_{iju}^{(s)})m_1 - \sum_{j\in\mathcal{J}} c_j^{(f)} y_j^{(heu)})$
20: Subject To Constraint 4.18
21: Solve Cplex Problem and Get Value For $y_j^{(heu)}$
22: Calculate New $x_{piju}^{(f)} = x_{piju}^{(f)} y_j^{(heu)}, \quad i \in \mathcal{I}, j \in \mathcal{J}, u \in \mathcal{U}$
23: Let $obj = 0$
24: $obj = \sum_{j\in\mathcal{J}} -c_j^{(f)} y_j^{(heu)}$
25: $obj = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{u\in\mathcal{U}}((p_{iju} - c_{iju}^{(v)})x_{piju}^{(f)})(r_{iju} + c_{iju}^{(p)} + c_{iju}^{(s)})m_1$
26: **Calculate Final** x_{iju} **and** y_j **Values:**
27: **If** $(obj \geq obj^{(new)})$ **Then** $(obj^{(new)} = obj$; $y_j = y_j^{(heu)}$; $x_{iju} = x_{piju}^{(f)})$ **end if**

28: **end for**
29: Provide Master Problem with x_{iju}, y_j, $c^{(d)}_{ijuq}$, h_{jiuqt} and o_{ijc} as starting solution

---

After generating results for both decision variables $x^{(f)}_{piju}$ and $y^{(heu)}_j$, the objective function value needs to be computed using the logic stated throughout lines 23 to 25. After applying the preceding logic, the result of the calculation is stored in variable obj. The final step in the heuristic algorithm is the execution of line 27. In this part of the algorithm, the value of obj is required. The objective value obj for each iteration is compared to the highest objective function value stored in obj^{new} as obtained from the previous iterations. When the heuristic function is initialised, both variables obj and obj^{new} are assigned values of 0. As the algorithm passes the first iteration, both obj and obj^{new} will take on the same value as the first objective function value as, at this stage, it will be greater than the initialisation value of 0. With the second iteration, obj could be less than obj^{new}, which would result in the logic encompassed in line 27 from being ignored. If obj has a greater value than obj^{new}, the model will enter the if statement in line 27. In this event, the previous objective value contained in variable obj^{new} will be replaced with the value contained in obj to ensure that obj^{new} always contains the highest calculated objective value. As part of updating obj^{new}, the algorithm will also assign $x_{iju} = x^{f}_{piju}$ and $y_j = y^{(heu)}_j$ so that the combination of binary decision variables used to calculate the new objective function is also stored for future use. When completing step 27 of the heuristic algorithm, the computational process re-enters the main permutation for loop, and the entire cycle is repeated for another permutation as contained in $funct[p^c][f^f]$. Given that parameter p^c is equal to 120, this main function, as described in the preceding section, will execute 120 iterations. At each iteration, decision variables x^{f}_{piju} and $y^{(heu)}_j$ will be computed. Suppose these variables can improve the objective function value greater than the previous highest stored value. In that case, the combination of binary values will be used to overwrite x_{iju} and y_j. At the end of the 120 iterations, the final state of x_{iju} and y_j together with decision variables $c^{(d)}_{ijuq}$, h_{jiuqt} and o_{ijc} will be fed to the column generation algorithm as an initial feasible solution to the product targeting problem.

As deliberated in the above section of the pseudocode (lines 9 and 12), the main heuristic makes use of a sub-function for the selection and sorting of $x^{(f)}_{piju}$. In order to perform the preceding operation, the function needs first to calculate the objective function value for each index i, j and u where $x^{(f)}_{piju}$ has been assigned a value of 1. During the first iteration of the main function, $x^{(f)}_{piju}$ has been assigned values of 1 for each index of p, i, j and u. As the algorithm moves through the various ranking and selection functions, some values of 1 contained in $x^{(f)}_{piju}$ are replaced with 0 to ensure that the results of x_{iju} adheres to the various upper and lower bounds of the optimisation constraints. More detail will be provided regarding the preceding in the discussions to follow. The objective function equation used within the sub-function is representative of constraint set (4.9) with the fixed product cost term ($\sum_{j\in\mathcal{J}} c^{(f)}_j y^{(heu)}_j$) being excluded from the objective function calculation as it does not vary with the changing values of decision variable $x^{(f)}_{piju}$. The objective function values calculated for each index i, j and u are stored in variable val_{iju} as per the logic noted in line 3 of the sub-function pseudocode. Variable val_{iju} is initialised to 0 in line 2 each time the sub-function is executed within the main heuristic framework.

After calculating the objective function value for each index, the objective values need to be sorted in descending order as part of the initial phases of the selection process. The logic designed for this sorting operation is represented in lines 4 to 15 of the secondary pseudocode. The sorting process consists of a main overarching for loop and two sub-for loops. The purpose of the main-for statement is to loop over the iterated index/indices of the constraint associated with the variable UB. As part of the main for loop, we define a vector pair called vp as seen in line 6. This vector is used to store the objective function value as well as the associated index when performing the sorting process. This step is imperative as the algorithm needs to track which index of $x^{(f)}_{piju}$ has the most significant influence on the objective function outcome in descending order. After defining vp, the algorithm

moves to the first sub-for statement, which is used to loop over the summated index/indices of the constraint associated with UB (lines 7 – 9). In this first sub for loop, the vector pair vp is populated with the actual objective function values and their associated indices. As the algorithm exists in the first sub-for loop, a sorting algorithm is applied as seen in line 10 of the pseudocode. The sorting algorithm is designed to sort the objective function value and its associated index in reverse order (descending), with the objective value being stored in the first integer index of the vector pair variable vp and the objective index in the second integer index of vp. The algorithm leverages the objective function value index to update the values of $x_{piju}^{(f)}$. The objective value is just used to sort the indices in the correct descending order so that the algorithm can use the sorted index list in the subsequent coding logic. When sorting has been completed, the algorithm progresses to the second sub for loop statement, which can be noted in lines 11 to 14. The primary purpose of the second sub-loop statement is to loop over the summated index/indices of the constraint associated with UB while substituting the value of val_{iju} with the sorted objective function values $vp.first$ and the value of val_{iju}^{ind} with the sorted objective function index values $vp.second$. The preceding sub-loop logic is utilised to extract the ordered objective index values that will be utilised throughout the pseudo logic seen in lines 16 – 26. After exiting the second sub-loop statement, the code enters the main-for statement. The above coding logic is executed again until the objective function sorting algorithm has completed its entire iteration cycle.

In lines 16 – 26 of the heuristic sub-function, binary values are assigned to temporary variable $x_{iju}^{(ft)}$ until the upper bound is satisfied for the constraint under consideration. The values assigned to temporary variable $x_{iju}^{(ft)}$ are used to calculate the end state value of $x_{piju}^{(f)}$ later throughout the heuristic algorithm. As part of the main loop computational logic for lines 16 – 26, the algorithm needs to loop over the iterated index/indices of the constraint associated with UB. When entering the main loop, variable UB is assigned the value of the left-hand side associated with the optimisation constraint being considered at the point in time. After assigning a value to variable UB, the heuristic enters a sub-for statement responsible for looping over the summated index/indices of the constraint associated with UB. The logic contained within the sub-loop is designed to perform binary decisions in order to determine the values which will be assigned to variable $x_{iju}^{(ft)}$. In line 20, we design an equation to track the delta value between the upper bound value (UB) of the constraint under consideration and the number of indices which has been assigned a value of 1 for variable $x_{iju}^{(ft)}$ (tracked by *counter*). The preceding is implemented in order to ensure that the algorithm does not exceed the constraint limits which govern the decision process when assigning values of 1 to $x_{iju}^{(ft)}$. In this way, the algorithm will always propose a feasible solution to the set of constraints being considered, as it will not be allowed to violate the bounds of the constraints. To enforce the preceding, lines 21 – 24 are incorporated into the heuristic algorithm. The if statement performs a validation check on variable UL. If variable UL contains a value greater than 0, the heuristic algorithm is allowed to assign a binary value 1 to $x_{iju}^{(ft)}$ where the index correspond to the value of val_{iju}^{ind}. It is, however, important to take note that when a value of 1 has been assigned to $x_{iju}^{(ft)}$, variable *counter* is incremented by order of 1. When *counter* equates to the upper bound value of UB, the heuristic algorithm will not be allowed to assign a value of 1 to $x_{iju}^{(ft)}$ for the remainder of the indices of val_{iju}^{ind} as to prevent the proposal of an infeasible combination for $x_{iju}^{(ft)}$. When UL reaches a value of 0, the algorithm is forced to exit the sub-loop logic, where after it enters the main loop logic. The computational process is repeated until the full iteration cycle is completed for the associated optimisation constraint.

The last phase of the sub-heuristic function is to update the value of $x_{piju}^{(f)}$ with the binary combinations contained in $x_{iju}^{(ft)}$. To update $x_{piju}^{(f)}$, the pseudocode reported in line 26 is executed. At the start of the main heuristic function, variable $x_{piju}^{(f)}$ is initialised using only 1 for each index p, i, j and u as stated in the previous section. This is to allow the algorithm to start from scratch when performing its decisions without introducing any pre-defined biases to the end state of the

variable $x^{(f)}_{piju}$. Using only values of 1 allows the model to perform selection across all permutations, customers, products, and channels, as none are excluded at the start of the heuristic algorithm. The first time variable $x^{(f)}_{piju}$ is multiplied with $x^{(ft)}_{iju}$, some of the values will automatically take on 0 as the number of positive selection (1) assignments to variable $x^{(ft)}_{iju}$ is governed by the upper bound limitations obtained from the optimisation constraints under consideration. When the execution of the heuristic sub-function is completed, the main algorithm is provided with a proposed value for $x^{(f)}_{piju}$. When more than one optimisation constraint is considered in the heuristic, which is the case in this thesis, the values proposed for $x^{(f)}_{piju}$ will change each time the heuristic sub-function code is executed. The coding logic encompassed in the sub-function will, however, ensure that the end state of $x^{(f)}_{piju}$ will adhere to every constraint being considered for the heuristic. Variable $x^{(f)}_{piju}$ will be sent through a series of ranking and selection functions, with each function reducing the number of positive selections being assigned to $x^{(f)}_{piju}$ until ultimately a feasible solution is obtained that adheres to all of the prescribed limitations.

The greedy starting heuristic algorithm was designed to allow for the dynamic specification of the number of constraints that should be considered when performing the selection process of x_{iju} and y_j. When considering the problem defined in this thesis, the heuristic algorithm is required to provide the primal master problem (MP_P) with a starting solution. Note that only 5 of the 8 constraints contained in the column generation master problem (Section 6.3.2) are considered when running through the heuristic algorithm. However, a feasible starting solution is still obtained. The reason why only some 8 constraints are incorporated is because of the greedy algorithmic approach which was implemented. The 3 constraints that are excluded from the heuristic model are lower bound constraints.

Algorithm 2: Heuristic Sub Function Pseudocode: $F(p, x^{(f)}_{piju}, UB)$

Calculate Objective Value Using $x^{(f)}_{piju}$
Let $x^{(ft)}_{iju} = 0$; $val_{iju} = 0$; $UB = 0$; $UL = 0$; $counter = 0$
Calculate objective function: $val_{iju} = ((p_{iju} - c^{(v)}_{iju})x^{(f)}_{piju})m_1(r_{iju} + c^{(p)}_{iju} + c^{(s)}_{iju}) \quad i \in \mathcal{I}, j \in \mathcal{J}, u \in \mathcal{U}$
Sort Objective Value and Matrix Index In Descending Order
loop over the iterated index/indices of the constraint associated to UB **then do**
    vector<pair<int,int> >vp
    loop over the summated index/indices of the constraint associated to UB **then do**
        vp.push_back(make_pair(val_{iju},index))
    end loop
    sort(vp.begin(),vp.end(),sortinrev)
    loop over the summated index/indices of the constraint associated to UB **then do**
        $val_{iju} = vp$.first
        $val^{ind}_{iju} = vp$.second
    end loop
end loop
Compute Temporary $x^{(ft)}_{iju}$ **Variable Until Upper Bound for Constraint Is Satisfied**
loop over the iterated index/indices of the constraint associated to UB **then do**
    $UB =$ LHS of Constraint
    loop over the summated index/indices of the constraint associated to UB **then do**
        $UL = UB - counter$
        if $(UL > 0)$ **then**
            ($x^{(ft)}_{iju} = 1$ for index val^{ind}_{iju})
            $counter$ += 1
        end if
    end loop
end loop
Update Variable $x^{(f)}_{piju}$
Update: $x^{(f)}_{piju} = x^{(ft)}_{iju} x^{(f)}_{piju} \quad i \in \mathcal{I}, j \in \mathcal{J}, u \in \mathcal{U}$
return $x^{(f)}_{piju}$

With a greedy approach, the algorithm tries to satisfy the maximum value of each constraint by

assigning binary values to $x_{iju}^{(ft)}$. By trying to meet the maximum criteria for each constraint, it will inherently also ensure that the proposed binary combination selected for $x_{iju}^{(ft)}$ will also satisfy the lower bound requirements. Therefore, leveraging the above phenomenon reduces the complexity and computational time of the starting heuristic algorithm while still ensuring a feasible initial solution is provided to the column generation method. Note that the heuristic algorithm only provides the user with a feasible solution to the optimisation problem but cannot generate a global optimal solution. The preceding is why the Dantzig-Wolfe decomposition and column generation algorithms are employed to reach global optimality. The foregoing mathematical framework is discussed in detail throughout Sections 6.3.2 to 6.3.7, with the initial focus being set on the primal master formulation for the product targeting optimisation problem.

6.3.2 Primal master problem

The next step in the product targeting column generation approach is to solve the primal master problem. The baseline optimisation model to be solved using the column generation algorithm is discussed in Section 4.3, known as the product targeting IP formulation (M_1). To transform the product targeting IP formulation model into a standard form that will allow for the use of the column generation algorithm, M_1 is reformulated using the Dantzig-Wolfe Decomposition algorithm for integer programming problems as elaborated on in Section 5.1. Regarding the Dantzig-Wolfe Decomposition theory, one needs to formulate the problem into a block angular structure to identify the linking and complicating constraints within the problem framework. The set of complicating constraints represents several optimisation problems that are independent of one another and will form the sub-problems within the column generation framework. The linking constraints refer to an optimisation model that involves decision variables from multiple sub-problems, which disrupts the problem's independent structure and cannot be solved independently. The preceding problem is the master problem within the column generation framework. Section 6.3.2 will focus on the linking constraints of M_1 and the reformulation of said constraints into the master problem used within the column generation framework.

6.3.2.1 Model objective

As an introduction to the primal master problem, consider the mathematical formulation of the objective function (6.1). This objective function takes on a similar mathematical form as noted in Section 4.3.1, with some minor transformations to align with the Dantzig-Wolfe Decomposition framework. To reformulate the objective function seen in Section 4.3.1 into the mathematical formulation (6.1) – (6.5), the objective function should be expressed as a convex combination of the extreme points for the various decision variables x_{iju}, $c_{ijuq}^{(d)}$, h_{jiuqt} and o_{ijc}. As mentioned at the start of this chapter, we introduce index $w \in \mathcal{W}$ to the optimisation model, which is used for tracking the number of extreme point combinations considered for the primal master problem. Linear decision variable z_{wj} is added to the problem formulation to track the real number weighting assigned by the optimisation problem to each extreme point. If z_{wj} is assigned a value of 0 for a given index w and j, it would mean that the extreme point is not being considered in the optimisation model and that an alternative extreme point exists which has a more significant influence in maximising the objective function value. When decision variable z_{wj} is assigned a value of 1, the associated extreme point is selected to take part in maximising the objective function value. Note that z_{wj} can also be assigned real values such as 0.5, which would mean that there is a convex combination of extreme points that has the best chance of maximising the problem objective function. When the column generation algorithm obtains a global optimal solution, only binary values will be present in z_{wj}. In order to solve the product targeting problem using column generation, we only consider a reduced master problem at the start of the column generation process. This means that the optimisation algorithm index w would take on a value of 1, representing only one extreme point being considered during model initialisation for each variable. This extreme point coincides with the starting solution obtained from the heuristic algorithm. The column generation master problem objective function is represented by MP_P.

The parameters utilised in the objective function (6.1) are identical to those considered in constraint set (4.8) with P_{wj} representing the monetary gain obtained from the product targeting process, CH_{wj} guiding the model to perform channel selection, TM_{wj} allowing the model to select the best time of day to call a customer and lastly, CR_{wj} which allows the model to account for cross-selling opportunities. The only difference in (6.1) when compared to constraint set (4.8) is the addition of index $w \in \mathcal{W}$ and decision variable z_{wj}.

$$(MP_P) \quad \max \quad \sum_{w \in \mathcal{W}} \sum_{j \in \mathcal{J}} z_{wj}(P_{wj} + CH_{wj} + TM_{wj} + CR_{wj}) \quad (= Z) \tag{6.1}$$

The purpose of (6.2) remained similar to what was discussed for constraint set (4.9). It is, however, imperative to note that in constraint set (4.9), variable x_{iju} was used as a decision variable, with its value being determined by the optimisation model. When considering (6.2), it should be noted that index w is added to x_{iju}, resulting in parameter x_{wiju} being obtained. Variable x_{iju} is changed from a decision variable to a parameter containing extreme point combinations of x_{iju}, which is represented by x_{wiju}. Meaning that x_{wiju} takes on predefined binary values (constants) which were either obtained from the starting heuristic solution (start of column generation algorithm) or from values generated by the sub-problem as specified in Section 6.3.4. The parameter x_{wiju} is multiplied with decision variable z_{wj} in order to allow the optimisation model to select the extreme point combinations contained in x_{wiju} corresponding to the global optimal solution.

The parameter y_j also takes on constant binary values as obtained from the starting heuristic. Note, however, that the value for y_j does not change as the code progresses through the column generation algorithm. The starting heuristic algorithm can obtain the optimal combination of products to consider, where after the column generation algorithm is required to use this combination of products to perform decisions on the remainder of the decision variables. For this reason no index w is added to y_j.

$$P_{wj} = \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} ((p_{iju} - c_{iju}^{(v)}) x_{wiju})(r_{iju} + c_{iju}^{(p)} + c_{iju}^{(s)}) m_1 - \sum_{j \in \mathcal{J}} c_j^{(f)} y_j, \quad w \in \mathcal{W}, j \in \mathcal{J} \tag{6.2}$$

Decision variables $c_{ijuq}^{(d)}$, h_{jiuqt} and o_{ijc} in constraint sets (4.10) – (4.12) are also augmented by adding index w to these variables in order to transform them to contain static extreme point combinations instead of binary variables that need to be calculated by the optimisation model as was done in Section 4.3. Similar to (6.2), the preceding extreme point parameters are multiplied with z_{wj} to allow the master problem to select the relevant extreme points for each parameter $c_{wijuq}^{(d)}$, h_{wjiuqt} and o_{wijc} which is associated to the global optimal solution. Except for the alterations above, the definition and purpose of (6.3) – (6.5) within the objective function remained similar to what was discussed for constraint sets (4.10) – (4.12).

$$CH_{wj} = \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} \sum_{q \in \mathcal{Q}_{iu}} (c_{ijuq}^{(q)} c_{wijuq}^{(d)}) m_2, \quad w \in \mathcal{W}, j \in \mathcal{J} \tag{6.3}$$

$$TM_{wj} = \sum_{i \in \mathcal{I}} \sum_{u=1} \sum_{q \in \mathcal{Q}_{iu}} \sum_{t \in \mathcal{T}} c_{iuqt}^{(t)} h_{wjiuqt}, \quad w \in \mathcal{W}, j \in \mathcal{J} \tag{6.4}$$

$$CR_{wj} = \sum_{i \in \mathcal{I}} \sum_{c \in \mathcal{C}_{ij}} c_{ijc}^{(r)} o_{wijc}, \quad w \in \mathcal{W}, j \in \mathcal{J} \tag{6.5}$$

Constraints (6.2) – (6.5) are lumped into a maximisation function (6.1), similar to what was done in Sections 4.2 and 4.3, in order to guide the optimisation model in maximising the financial institution's monetary gain and allowing the selection of the best conversation at the right time through the correct channel to improve customer satisfaction. The decisions performed in the master problem

objective function are governed by the linking constraints defined within the product targeting problem discussed in Section 4.3. An overview of the mentioned constraints is provided in Section 6.3.2.2, with a detailed discussion of how the constraints had to be altered to adhere to the column generation framework.

6.3.2.2 Model constraints

The linking constraints had to be identified from the list of constraints reported in Section 4.3.2 to extract constraint set for the column generation master problem. The criteria used to determine if a constraint falls within the linking constraint category was to identify if the constraint can be split up per product index j. If the relevant constraint requires a full view of the entire product list in order to maintain its original purpose (i.e., the constraint requires a summation of variables across the product offerings), it would mean that the constraint cannot be split up per product index j. Therefore it will be classified as a linking constraint. However, it is imperative to note that the preceding theory does not apply to constraint sets (6.10) – (6.11), and some modeling adjustments had to be made to include the mentioned constraints in the master problem formulation. The purpose of constraint sets (6.6) – (6.9) remained similar to constraint sets (4.13), (4.15), (4.20), and (4.21), with the only changes being the addition of the column generation decision variable z_{wj} and changing decision variable x_{iju} to x_{wiju} for it to contain static extreme point combinations instead of binary values which need to be determined by the optimisation model. Index $w \in \mathcal{W}$ is also added to account for the various extreme point combinations that are appended to the problem formulation as the model progresses through the column generation iterations. No other alterations had to be made to constraint sets (6.6) – (6.9) to incorporate them into the column generation master problem framework.

$$\sum_{w\in\mathcal{W}}[\sum_{j\in\mathcal{J}} z_{wj}(\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}} p_{iju}x_{wiju} - R(\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}} c^{(v)}_{iju}x_{wiju} + c^{(f)}_{j}y_j))] \geq 0 \tag{6.6}$$

$$\sum_{w\in\mathcal{W}}\sum_{j\in\mathcal{J}} z_{wj}(\sum_{u\in\mathcal{U}} x_{wiju}) \leq (1 - r^{(c)}_{i}), \quad i \in \mathcal{I} \tag{6.7}$$

$$\sum_{w\in\mathcal{W}}\sum_{j\in\mathcal{J}} z_{wj}(\sum_{i\in\mathcal{I}} x_{wiju}) \geq c^{(l)}_{u}, \quad u \in \mathcal{U} \tag{6.8}$$

$$\sum_{w\in\mathcal{W}}\sum_{j\in\mathcal{J}} z_{wj}(\sum_{i\in\mathcal{I}} x_{wiju}) \leq c^{(m)}_{u}, \quad u \in \mathcal{U} \tag{6.9}$$

As mentioned in the subsequent discussion, constraint sets (6.10) – (6.11) do not fit the standard criteria used to identify constraints that should enter the master problem formulation. Note that these constraints do not require the summation of variables across product indices j to maintain its original purpose. Instead, it is focused on iterating through each product j to ensure that the upper and lower limits hold. The original constraints before transformation can be noted in constraint sets (4.22) – (4.23). In order to add the preceding constraints into the master problem, we first moved the right-hand side of the constraints to the left resulting in constraint set (4.22) being equal to or less than 0 whereas constraint set (4.23) is set equal to or greater than 0. After reordering the variables, the left-hand side is multiplied by z_{wj} to allow the model to select the best extreme point combination x_{wiju} while adhering to the limitations of constraint sets (6.10) – (6.11). Rewriting the mentioned constraints into the portrayed format allowed it to be incorporated into the master formulation. Note, however, that constraint sets (6.10) – (6.11) could also have been excluded from the master problem and added to the sub-problem formulation seen in Section 6.3.4. However, to reduce the complexity of the sub-problems, it was decided to incorporate the said constraints into

the master problem formulation. Given that the branch-and-bound algorithm is used to solve the sub-problem, we want to limit the complexity to prevent computational memory limitations.

$$\sum_{w\in\mathcal{W}} z_{wj}(\sum_{i\in\mathcal{I}} x_{wiju} - (1-E_{ju})l_{ju}^{(m)}) \leq 0, \quad j\in\mathcal{J}, u\in\mathcal{U} \tag{6.10}$$

$$\sum_{w\in\mathcal{W}} z_{wj}(\sum_{i\in\mathcal{I}} x_{wiju} - (1-E_{ju})l_{ju}^{(n)}y_j) \geq 0, \quad j\in\mathcal{J}, u\in\mathcal{U} \tag{6.11}$$

No significant alterations were made to constraint set (4.24) to derive constraint set (6.12). Only changes were to add index w, include z_{wj} and update decision variable x_{iju} to x_{wiju}. The purpose of constraint set (6.12) within the column generation context remained similar to what was discussed for constraint set (4.24).

$$\sum_{w\in\mathcal{W}}\sum_{j\in\mathcal{J}} z_{wj}x_{wiju} \leq m_3(1-c_{iu}^{(e)}), \quad i\in\mathcal{I}, u\in\mathcal{U} \tag{6.12}$$

Variable z_{wj} is added to the left-hand side of constraint set (4.18) in order to obtain constraint set (6.13). Note that z_{wj} will be assigned a value of 0 for index j when a product is not considered throughout the decision process of the optimisation model. The parameter y_j is excluded from constraint set (6.13) as z_{wj} will perform a similar task within this constraint as was managed by y_j in constraint set (4.18).

$$\sum_{w\in\mathcal{W}}\sum_{j\in\mathcal{J}} z_{wj} \leq P^{(m)} \tag{6.13}$$

As part of the Dantzig-Wolfe Decomposition reformulation, it is required that constraint sets (6.14) and (6.15) are added to the column generation master problem to maintain model feasibility. The purpose of constraint set (6.14) is to ensure that the fractional values assigned to z_{wj} within the master optimisation model do not exceed an overall value of 1 per product index j. If z_{wj} is allowed to exceed 1 for a given index j, it would not make sense as the extreme points associated with index j would then violate the definition of a convex combination. Variable z_{wj} should thus maintain a convex combination of the extreme points provided to the master problem, and for this reason, the summation of index w for this variable should be curbed at a value of 1. In constraint set (6.14), a less than or equal to sign is used instead of a pure equal to sign as dictated by the Dantzig-Wolfe Decomposition Algorithm. This is to allow the column generation algorithm to assign fractional values to z_{wj} during the start of the computational process. As the algorithm moves through the computations, the values of z_{wj} will start to converge to a value of 1. At the point where the optimisation model terminates at a global optimal solution, z_{wj} will have been assigned a value of either 0 or 1 for each given index j.

$$\sum_{w\in\mathcal{W}} z_{wj} \leq 1 \quad j\in\mathcal{J} \tag{6.14}$$

Lastly, constraint set (6.15) is added to the master problem in order to allow the optimisation model to assign real values to z_{wj} with the values being constrained to a lower limit of 0 and an upper limit of 1. The reason why z_{wj} cannot be limited to binary decision variables is that the primary master problem makes use of the Simplex method as an algorithm to compute the values for z_{wj}. The simplex method was designed to use linear programming problems and could not handle integer or binary problems. It is for this reason that z_{wj} takes on real values in the model formulation represented by constraint set (6.15).

$$0 \leq z_{wj} \leq 1, \quad w\in\mathcal{W}, j\in\mathcal{J} \tag{6.15}$$

The reason why the column generation algorithm makes use of the simplex method is in order to extract dual values as well as the reduced cost for the primal master problem so that the associated

sub-problem objective functions can be formulated. Algorithms such as the branch-and-bound would not allow the user to extract the preceding values from the algorithms and, therefore, would prevent the correct formulation of the relevant sub-problems required for the column generation process. Therefore, applying the Dantzig-Wolfe Decomposition algorithm is essential to transform the original binary optimisation model into a linear programming problem so that the simplex method can be applied within the column generation framework. In Section 6.3.3, the dual master problem derivation is utilised to calculate the reduced cost required for the sub-problem objective functions.

6.3.3 Dual master problem

The dual master problem is not explicitly used within the column generation framework. However, the problem's derivation is required to extract the reduced cost for the primal master problem. The theory of deriving the dual formulation from a primal problem has been discussed in detail throughout Chapter 5, with that theory being practically applied throughout Section 6.3.3 of the thesis. The derivation of the dual objective function is discussed in Section 6.3.3.1, whereas the dual problem constraints are presented in Section 6.3.3.2.

6.3.3.1 Model objective

The dual master problem objective function (MP_D) is represented by (6.16) – (6.18). The term M_1 within (6.16) is representative of the objective function terms derived from the first 4 constraints within the primal master problem. The term M_2 contains the objective function terms for the last 5 constraints noted in the primal master problem. Both terms M_1 and M_2 are lumped into a minimisation function, as seen in (6.16), to generate the final dual objective function. As part of the dual derivation theory, the dual master problem takes on the opposite operation of the primal master problem. If the primal master problem is a maximisation problem, then the dual problem will be a minimisation problem, as shown in (6.16).

$$(MP_D) \quad \min \quad (M_1 + M_2) \quad (= D) \tag{6.16}$$

The first term of M_1 is derived from the right-hand side value contained in (6.6), which is multiplied with dual value $d^{(a)}$. The dual value $d^{(a)}$ corresponds to the dual value obtained from (6.6) after solving the master problem using the simplex algorithm. Given that the right-hand side value equates to 0, the first term will not influence the overall model computations. However, from a mathematical derivation perspective, term one is included in (6.17) for completeness. The remainder of the terms in (6.17) is derived from the right-hand side values of (6.7) – (6.9) and its associated duel values $d_i^{(b)}$, $d_u^{(c)}$ and $d_u^{(d)}$ respectively.

$$M_1 = d^{(a)} + \sum_{i \in \mathcal{I}} (1 - r_i^{(c)}) d_i^{(b)} + \sum_{u \in \mathcal{U}} c_u^{(l)} d_u^{(c)} + \sum_{u \in \mathcal{U}} c_u^{(m)} d_u^{(d)} \tag{6.17}$$

The same methodology has been applied to derive the objective function terms for (6.18) as mentioned for (6.17). Note that the right-hand side terms for (6.10) – (6.14) were taken into consideration where after each term was multiplied with its associated dual values $d_{ju}^{(e)}$, $d_{ju}^{(f)}$, $d_{iu}^{(g)}$, $d^{(h)}$ and $d^{(k)}$ respectively in order to obtain the dual objective values as seen in M_2.

$$M_2 = d_{ju}^{(e)} + d_{ju}^{(f)} + \sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} m_3 (1 - c_{iu}^{(e)}) d_{iu}^{(g)} + P^{(m)} d^{(h)} + 1 d^{(k)} \tag{6.18}$$

Constraints (6.16) – (6.18) represent the dual objective function derived from the primal master problem. It is, however, imperative to note that the dual objective function is not leveraged within the column generation framework and therefore is not used in the subsequent algorithmic derivations of this thesis. The crucial mathematical derivation required for the column generation algorithm is the dual constraints, as these will be leveraged to formulate the reduced cost for each of the various sub-problems, as discussed in Section 6.3.3.3. The derivation of the dual constraints is provided in detail throughout Section 6.3.3.2.

6.3.3.2 Model constraint derivation

The formulation of the dual constraint consists of the term LHS_f (6.21), which is derived from the left-hand side of constraint sets (6.6) – (6.14) multiplied with their respective dual values. Decision variable z_{wj} is excluded from the derivation of LHS_f as this variable is only applicable to the primal master problem formulation. The equation portrayed for LHS_f is written in a reduced mathematical form to simplify the representation thereof. The term LHS_f is calculated by summating over the transformed left-hand sides of constraint sets (6.6) – (6.14) as mentioned above.

$$LHS_1 = x_{iju}(d_i^{(b)} + d_u^{(c)} + d_u^{(d)} + d_{ju}^{(e)} + d_{ju}^{(f)} + d_{iu}^{(g)}) \tag{6.19}$$

$$LHS_2 = \sum_{j\in\mathcal{J}}\sum_{u\in\mathcal{U}}[E_{ju}l_{ju}^{(m)}d_{ju}^{(e)} + E_{ju}l_{ju}^{(n)}y_j d_{ju}^{(f)} - 2] + \sum_{j\in\mathcal{J}}[-c_j^{(f)}Rd^{(a)}y_j + d^{(h)} + d_j^{(k)}] \tag{6.20}$$

$$LHS_f = \sum_{j\in\mathcal{J}}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}[(p_{iju} - Rc_{iju}^{(v)})d^{(a)}x_{iju} + LHS_1] + LHS_2 \tag{6.21}$$

To construct the right-hand side equation of the dual constraint as denoted by term RHS_f (6.24), the primal master problem objective function values are utilised. There are no additions of dual values or alterations to the mathematical equations of the objective function values (6.2) – (6.5) in order to derive constraint set (6.24). The term RHS_f is only constructed by summating (6.2) – (6.5) into one equation.

$$RHS_1 = \sum_{j\in\mathcal{J}}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}(p_{iju} - c_{iju}^{(v)})(r_{iju} + c_{iju}^{(p)} + c_{iju}^{(s)})m_1 x_{iju} - \sum_{j\in\mathcal{J}} c_j^{(f)}y_j \tag{6.22}$$

$$RHS_2 = \sum_{j\in\mathcal{J}}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\sum_{q\in\mathcal{Q}_{iu}}(c_{ijuq}^{(q)}c_{ijuq}^{(d)})m_2 + \sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}}\sum_{u=1}\sum_{q\in\mathcal{Q}_{iu}}\sum_{t\in\mathcal{T}} c_{iuqt}^{(t)}h_{jiuqt} \tag{6.23}$$

$$RHS_f = RHS_1 + RHS_2 + \sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}}\sum_{c\in\mathcal{C}_{ij}} c_{ijc}^{(r)}o_{ijc} \tag{6.24}$$

The final condensed dual constraint is portrayed by constraint set (6.25) with term LHS_f in constraint set (6.21) being set to greater than or equal to term RHS_f in constraint set (6.24). The preceding, forms the baseline for the computation of the reduced cost equation, which will be used to construct the sub-problem objective functions required in the column generation algorithm.

$$LHS_f \geq RHS_f \tag{6.25}$$

In section 6.3.3.3, the insights generated from the above derivation of the dual constraint are used to construct the reduced cost equation.

6.3.3.3 Reduced cost

The difference is taken between LHS_f and RHS_f as seen in (6.26) to compute the reduced cost for the product targeting sub-problem. By substituting the values of LHS_f and RHS_f as computed in (6.21) and (6.24) into the left-hand side of (6.26) and rewriting the mathematical formulation in it's simplest form, the reduced cost equation is derived as (6.31).

$$LHS_f - RHS_f \leq 0 \tag{6.26}$$

$$LHS_1 = [p_{iju}(d^{(a)} - (r_{iju} + c_{iju}^{(p)})m_1) - c_{iju}^{(v)}(Rd^{(a)} + (r_{iju} + c_{iju}^{(p)} + c_{iju}^{(s)})m_1)] \tag{6.27}$$

$$LHS_2 = \sum_{j\in\mathcal{J}}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}} x_{iju}[LHS_1 + [d_i^{(b)} + d_u^{(c)} + d_u^{(d)} + d_{ju}^{(e)} + d_{ju}^{(f)} + d_{iu}^{(g)}]] \tag{6.28}$$

$$LHS_3 = \sum_{j\in\mathcal{J}}[c_j^{(f)} y_j(1 - Rd^{(a)}) + d^{(h)} + d_j^{(k)}] + \sum_{j\in\mathcal{J}}\sum_{u\in\mathcal{U}}[E_{ju} l_{ju}^{(m)} d_{ju}^{(e)} + E_{ju} l_{ju}^{(n)} y_j d_{ju}^{(f)} - 2] \tag{6.29}$$

$$LHS_4 = -\sum_{j\in\mathcal{J}}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\sum_{q\in\mathcal{Q}_{iu}} (c_{ijuq}^{(q)} c_{ijuq}^{(d)}) m_2 - \sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}}\sum_{u=1}\sum_{q\in\mathcal{Q}_{iu}}\sum_{t\in\mathcal{T}} c_{iuqt}^{(t)} h_{jiuqt} \tag{6.30}$$

$$R^C = LHS_2 + LHS_3 + LHS_4 - \sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}}\sum_{c\in\mathcal{C}_{ij}} c_{ijc}^{(r)} o_{ijc} \tag{6.31}$$

After computing the reduced cost value noted in (6.31), the final simplified dual constraint can be represented as (6.32), with the reduced cost being set to less than or equal to zero.

$$R^C \leq 0 \tag{6.32}$$

As part of the column generation algorithm, the derived reduced cost is used to determine when the algorithm has obtained an answer close to or equal to the global optimal solution. The preceding methodology is derived from the simplex algorithm. When considering a minimisation sub-problem, a negative reduced cost (objective function value) would indicate the model's ability to generate alternative columns, which could still improve the overall master problem solution. As per the column generation algorithm, the model computations will not terminate in such a scenario. Instead, the model will enter into another iteration to try and determine a better extreme point combination that would maximise the master problem value. If the reduced cost computed for the sub-problem reaches a positive value (≥ 0), it would indicate that the model could calculate an extreme point combination capable of generating a global optimal solution. In this scenario, the column generation algorithm will terminate as the proposed solution could not be improved using additional column generation iterations.

The reduced cost (6.31) will be implemented as the objective function equation for the column generation sub-problem. The reduced cost will initially start with a very large negative value (for the minimisation problem under consideration), where after, the value will start to converge towards 0 as the columns added to the optimisation model improves the master problem objective function. For realistically sized optimisation problems, when the column generation method starts to converge, the algorithm must run through a multitude of iterations to induce a marginal change in the master and sub-problem objective values. This could result in extensive computational time required if one wants to use the reduced cost to prove global optimality (waiting until the reduced cost reaches a positive value). As a result, we not only use reduced cost to determine global optimality but compute performance metrics such as the optimality and integrality gap to guide the solution, as reported in Section 6.3.6. More detail regarding the computational time required to obtain a solution will be discussed in Chapter 8.

6.3.4 Sub-problem

The sub-problem, also known as the pricing problem, which is used within the column generation algorithm, consists of an objective function that is derived from the dual master problem reduced cost as well as model constraints that coincide with the complicating constraints identified from the IP model presented in Section 4.3.1. Section 6.3.4 focuses on the sub-problem objective function, where after a detailed analysis is provided on the model constraints. In Section 6.2, the theory of parallel processing was introduced. Given the structure of the product targeting column generation

sub-problem, the overarching integer programming problem can be subdivided into multiple smaller integer programming problems to allow parallel processing to speed up solution time. When evaluating the mathematical structure, we note that the optimisation problem can be split up by index $j \in \mathcal{J}$ as none of the model constraints are summated across index j. This means that we can create j number of small optimisation models, and solve them in parallel, to improve the computational time of the overarching column generation algorithm. Details are provided regarding the improvement in computational efficiency by splitting the sub-problem into multiple smaller optimisation models in Chapter 8. Note that the sub-problem(s) are solved using the standard branch-and-bound method, given that these problems are constructed as integer programming models.

6.3.4.1 Model objective

The reduced cost derived from the dual master problem is the objective function for the column generation sub-problem. Note that the reduced cost is lumped into a minimisation function to complete the objective function's derivation. Refer to (6.33) for the mathematical formulation of the preceding.

$$(SP) \quad \min \quad (R^C) \tag{6.33}$$

When the primal master problem is a maximisation problem, the associated sub-problem will take on the form of a minimisation problem. The opposite also holds when the primal master problem is considered a minimisation problem. This is why the sub-problem is defined as a minimisation problem for this study.

6.3.4.2 Model constraints

The column generation sub-problem constraints are represented by constraint sets (6.34) – (6.43). Note that these constraints are extracted from the IP model seen in Section 4.3.2 (from the list of complicating constraints), with no alterations being made to incorporate them into the column generation sub-problem framework. The only change required was to pair constraint sets (6.34) – (6.43) to the correct objective function (6.33) in order to generate the required sub-problem outputs. Note that the purpose of these constraints within the sub-problem also remains identical to what was discussed in Section 4.3.2, and therefore no additional information is needed. The constraints in Section 4.3.2 corresponding to constraint sets (6.34) – (6.43) are as follow: (4.14), (4.16), (4.17), (4.25), (4.19), (4.26), (4.27), (4.28) and (4.29).

$$\sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} c_{ij}^{(v)} x_{iju} \leq B_j, \quad j \in \mathcal{J} \tag{6.34}$$

$$\sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} x_{iju} \geq l_j^{(l)} y_j, \quad j \in \mathcal{J} \tag{6.35}$$

$$\sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} x_{iju} \leq l_j^{(u)} y_j, \quad j \in \mathcal{J} \tag{6.36}$$

$$c_{ijuq}^{(d)} \leq (1 - c_{ijuq}^{(pr)}), \quad j \in \mathcal{J}, i \in \mathcal{I}, u \in \mathcal{U}, q \in \mathcal{Q}_{iu} \tag{6.37}$$

$$\sum_{u \in \mathcal{U}} x_{iju} \leq 1, \quad j \in \mathcal{J}, i \in \mathcal{I} \tag{6.38}$$

$$\sum_{q \in \mathcal{Q}_{iu}} c_{ijuq}^{(d)} = x_{iju}, \quad j \in \mathcal{J}, i \in \mathcal{I}, u \in \mathcal{U} \tag{6.39}$$

$$\sum_{q\in\mathcal{Q}_{iu}}\sum_{t\in\mathcal{T}} h_{jiuqt} \leq m_4 x_{iju}, \quad j\in\mathcal{J}, i\in\mathcal{I}, u=1 \tag{6.40}$$

$$\sum_{t\in\mathcal{T}} h_{jiuqt} \leq c^{(d)}_{ijuq}, \quad j\in\mathcal{J}, i\in\mathcal{I}, u=1, q\in\mathcal{Q} \tag{6.41}$$

$$\sum_{c\in\mathcal{C}_{ij}} o_{ijc} \leq \sum_{u\in\mathcal{U}, c^{(s)}_{iju}=1} c^{(s)}_{iju} x_{iju}, \quad j\in\mathcal{J}, i\in\mathcal{I} \tag{6.42}$$

$$x_{iju}, h_{jiuqt}, c^{(d)}_{ijuq}, o_{ijc} \in \{0,1\}, \quad j\in\mathcal{J}, i\in\mathcal{I}, u\in\mathcal{U}, q\in\mathcal{Q}_{iu}, t\in\mathcal{T}, c\in\mathcal{C}_{ij} \tag{6.43}$$

Note that the above constraints are representative of the overarching sub-problem. To apply parallel processing as elaborated at the start of Section 6.3.4, we split the above formulation into multiple sub-problems according to the number of products $j \in \mathcal{J}$. For example, if $\mathcal{J}$ takes on a value of 10 which would represent 10 products being considered for commitment in the product targeting problem, then the sub-problem could be divided into 10 smaller problems. The purpose of the multiple sub-problems is to determine the correct binary combination for decision variables x_{iju}, h_{jiuqt}, $c^{(d)}_{ijuq}$ and o_{ijc} that will decrease the reduced cost objective function value and in turn improve the master objective value. When an optimal binary combination is obtained, it will be fed to the master problem parameters x_{wiju}, h_{wjiuqt}, $c^{(d)}_{wijuq}$ and o_{wijc} in Section 6.3.2 as an additional extreme point to consider for decision. Answers generated from the sub-problems will be appended to the master problem as additional columns. New columns will be appended to the master problem until the algorithm reaches its termination point. As mentioned in Section 6.3.3.3, we use gap calculations to determine the point of termination for the column generation algorithm. In order to compute the various gap metrics, the algorithm first needs to calculate an upper bound for the master problem. In Section 6.3.5, the derivation of such an upper bound is provided.

6.3.5 Upper bound formulation

In order to compute the upper bound for the master problem, the weak duality theorem is used as a point of departure.

Consider a primal problem of the form:

$$maximize \quad c^T x \quad subject \quad to \quad Ax \leq b, \quad x \geq 0. \tag{6.44}$$

With the dual problem derived from the primal problem taking on the form:

$$maximize \quad b^T y \quad subject \quad to \quad A^T y \geq c, \quad y \geq 0. \tag{6.45}$$

If $(x_1, x_2 ... x_n)$ is taken as a feasible solution to the primal maximization linear program and $(y_1, y_2 ... y_n)$ is assumed to be a feasible solution to the dual minimization linear problem, then the weak duality theorem can be stated as follow:

$$c^T x = x^T c \leq x^T A^T y \leq b^T y, \tag{6.46}$$

$$\sum_{j\in\mathcal{J}} c_j x_j \leq \sum_{i\in\mathcal{I}} b_i y_i. \tag{6.47}$$

where c_j and b_i refers to the coefficients of the respective objective functions. The above statement alludes to the fact that the objective function value for the dual optimisation problem should be either greater than or equal to the primal objective function value when computed by the solution algorithm. Simplifying the preceding function, the subsequent equation is obtained:

$$\sum_{j\in\mathcal{J}} c_j x_j - \sum_{i\in\mathcal{I}} b_i y_i \leq 0. \tag{6.48}$$

6.3.6 Gap calculation

The optimality and integrality gaps are computed during each iteration of the solution framework to track the progress of the column generation algorithm in finding a global optimal solution. The optimality gap provides the end user with information regarding how far the obtained solution is from the global optimal solution. In contrast, the integrality gap guides the user on the fractional relationship between the upper bound and the master problem integer solution.

6.3.6.1 Optimality gap

In order to compute the optimality gap, we define an optimisation objective function MP_I which corresponds to the objective value MP_P when the decision variable z_{wj} is assigned binary values instead of real values. In other words, MP_I denotes the integer solution objective function for the linear programming problem MP_P.

$$MP_I = MP_P, \quad where \quad z_{wj} \in \{0,1\}, \quad w \in \mathcal{W}, j \in \mathcal{J} \tag{6.49}$$

Once MP_I has been determined, the optimality gap O^G can be computed. In (6.50) the difference is taken between U^B and MP_I where after it is divided by U^B and multiplied by 100 to convert the value to percentage.

$$O^G = ((U^B - MP_I)/U^B)100 \tag{6.50}$$

When executing the column generation algorithm, a solution termination point that fell within an optimality gap of less than 10% was considered as a close enough solution to global optimality. While solving the various test cases, which will be discussed in Chapters 7 – 9, it was identified that once the column generation algorithm crosses the 10% threshold for most problems, it enters the tailing-off effect. Meaning, significant amount of processing time and iterations are required to marginally improve the objective function value. To prevent this excessive compute time, a cut-off of 10% was chosen if the algorithm is able to reach this % optimality gap within a reasonable time. It is, however, essential to note that given the size and complexity of the problems being solved, it is sometimes required to terminate the optimisation problem prematurely (before it reaches $O^G \leq 10\%$). For these type of problems, it becomes acceptable to use additional parameters such as run time and resource consumption as secondary termination triggers to allow the end user to obtain a reasonable solution close to global optimality. Even when the algorithm might not reach global optimality in some cases, the optimality gap provides a good idea of how far from global optimality the proposed solution is.

6.3.6.2 Integrality gap

The algorithm has included the integrality gap in determining how far the proposed solution is from an integral solution. Note that this metric has not been used to make any model termination decisions and has only been included in the end user's interest. The integrality gap I^G is computed by dividing U^B with MP_I as seen in (6.51).

$$I^G = U^B/MP_I \tag{6.51}$$

As the algorithm progresses through the various iterations, it is expected that I^G will start to converge to a value of 1. At the start of the algorithm, I^G will contain a large positive value, but as the solution proposed by the column generation algorithm improves, I^G will start to move towards 1. The closer I^G gets to 1, the closer the algorithm gets to reaching global optimality.

6.3.7 Dealing with heading-in and yo-yo phenomenon

As discussed in detail throughout Chapter 5, aspects such as dual variable heading-in, yo-yo, and the tailing-off phenomenon could have a detrimental effect on the computational performance of the column generation algorithm. In order to reduce the initial heading-in effect experienced by the

column generation algorithm, we introduce (6.52) into the sub-problem objective function. A large constant value (*const*) is multiplied with a dynamic fraction value $(1 - a)$ with a being initialised at a value of 0. As the algorithm progresses through the iterations, a is incremented by 0.05 to reduce the effect (6.52) has on the sub-problem computations. The incremental value of 0.05 can be reduced or increased depending on the desired rate at which the end user would want (6.52) to influence the sub-problem objective function. After completing a certain amount of iterations, the value of a will equate to 1 (as per the incremental increases made to a), leaving the entire term $((1 - a)(const))$ to take on a value of 0. With (6.52) taking on a value of 0, it will no longer affect the column generation algorithm. The purpose of (6.52) is to force the algorithm out of the initial heading-in effect to allow the creation of sensible columns. The preceding methodology works when evaluating the modeling results presented in Chapter 8. However, after reducing the influence of the heading-in effect using (6.52), the algorithm moves towards the dual value yo-yo phenomenon.

$$(1 - a)(const) \tag{6.52}$$

The yo-yo phenomenon is caused by sporadic changes in the dual values calculated from the primal master problem. A sudden abrupt change in the dual values could result in the column generation algorithm frequently changing the direction in which the solution has been steered. As a result, the model could struggle to find appropriate columns which could improve the master problem objective function. To try and counteract the preceding, we introduce a dual smoothing algorithm (dual averaging) into the sub-problem objective function to try and dampen the influence of rapidly changing dual values. The proposed solution allows for changes in the dual values. However, it is managed in a more controlled manner to allow the algorithm to generate better columns much faster. The proposed solution is stipulated below.

At the start of the column generation algorithm, variable $d_{ave}^{(a-k)}$ is created for each of the dual values present in the sub-problem objective function. Variables $d_{ave}^{(a-k)}$ will take on the values of the initial iteration dual values $d^{(a-k)}$ as denoted in (6.53).

$$d_{ave}^{(a-k)} = d^{(a-k)} \tag{6.53}$$

As the algorithm moves to the next iterations, the column generation method starts to leverage (6.53) to calculate the average dual values across the various iterations. In (6.54), $d_{ave}^{(a-k)}$ is updated with the average value between $d_{ave}^{(a-k)}$ and $d^{(a-k)}$.

$$d_{ave}^{(a-k)} = (d_{ave}^{(a-k)} + d^{(a-k)})/2 \tag{6.54}$$

After computing the average dual values for a given point in time, we formulate an equation (6.55) that will consider both the current and the average dual values when determining the next column to enter the master problem. The model will consider the actual and average dual values in a fractional relationship. The preceding is enforced by introducing fractional variable a_1 into (6.55). At the start of the column generation algorithm variable a_1 will be initialised at an arbitrary value of say 0.8 (a_1 can be initialised with any value between 0 and 1). The preceding would mean that when the sub-problem considers the various dual values within the objective function, the actual dual values would have a 0.8 weighted influence on the objective function calculations. In contrast, the average dual values would have a 0.2 weighted influence. Variable a_1 would maintain a constant value of 0.8 throughout the column generation iterations. Only when the integrality gap reaches a point less than 1.1 will the value of a_1 be updated to 1. When a_1 takes on a value of 1, it will result in the dual average effect induced by $d_{ave}^{(a-k)}$ to be excluded from the model. With the exclusion of $d_{ave}^{(a-k)}$, the sub-problem objective function would only be influenced by $d^{(a-k)}$. Note that the integrality gap will only reach a point below 1.1 close to the end of the algorithm (small optimality gap), where we would not want the dual averages to induce biases on the columns being generated. For this reason, a_1 is then assigned a value of 1. However, before reaching that cut-off point of $I^G \leq 1.1$, the dual averaging technique (6.55) assists with reducing the yo-yo effect within the column generation algorithm.

$$d_f^{(a-k)} = d^{(a-k)}(a_1) + d_{ave}^{(a-k)}(1 - a_1) \tag{6.55}$$

By substituting each of the dual values seen in the reduced cost function (Section 6.3.3.3) with its dual smoothing equivalent calculated in (6.55), the computational efficiency of the column generation algorithm is significantly improved. The preceding dual smoothing methodology allows the column generation algorithm to reduce both computational time and memory requirements for the column generation algorithm. In Chapter 8, some tangible results are provided, emphasising the significant computational improvements induced by the implementation of both (6.52) and (6.55).

6.3.8 Removing unused columns from algorithm

As deliberated in Chapter 3, it is known that the column generation algorithm consumes a starting solution to initialise the computational process. Once initialised, both the master and sub-problems are solved iteratively. After solving the relevant sub-problems, the algorithm identifies columns capable of improving the master problem solution. At the point where the mentioned columns are identified, it is appended to the master problem. Note that each time the sub-problems are solved, new columns are identified and added to the master problem framework to form part of the computational process. As it stands, there will be columns added to the master problem early on in the iterative process, which is inferior to the more recent columns. In such a case, the simplex method would assign decision variable z_{wj} values of 0 for the inferior columns but still keep those columns in the cached memory as the algorithm moves through the various iterations. In doing this, the memory consumed by the model increases with every iteration. In order to optimise the preceding process, code was designed to evaluate the values assigned to z_{wj} at each iteration. When z_{wj} is assigned a value of 0 for a specific index w and j, the associated index is removed from the problem's memory framework. In doing this, the model only maintains a combination of z_{wj} variables, which partakes in the solution of the master problem and discards those that do not have any influence. By virtue, this reduces the problem's memory size quite significantly and speeds up the computational process. Even though there are some positives to this methodology, it was identified that this method would only be able to provide the end user with an approximate solution and will not be able to reach global optimality. The reason being is that even though the column generation algorithm assigns 0 values to z_{wj} at times, these 0 values still influence the computation of the following columns even if it does not contribute to the actual master problem objective function value. Therefore, removing these 0 columns changes the algorithm's computational dynamics, and in doing so, the algorithm cannot compute all of the necessary columns required to reach global optimality. This solution framework is, however, something that was tested for large, loosely constrained product targeting problems where normal algorithms struggle to find a close approximate solution or even fail to do so. In the preceding scenario, this solution framework can provide the end user with an approximate solution, even if it is not to the point of global optimality. Chapter 8 provides computational results where the mentioned solution framework is tested and implemented.

6.3.9 Novel column generation model contributions

A summarised list of all the contributions made to the product targeting column generation framework is provided in Section 6.3.9. The below list is primarily focused on the modeling and algorithmic contributions made throughout this study to allow end users to solve larger and more complex product-targeting optimisation problems that were not previously possible when using standard solution algorithms. The mentioned list is represented below:

1. Reformulation of the proposed complex product targeting problem proposed in Section 4.3 using Dantzig-Wolfe Decomposition and Column Generation theories,

2. Proposing a starting heuristic algorithm. The algorithm enables the end user to find a starting solution within reasonable bounds of the optimal solution for larger product targeting problems,

3. Derivation and implementation of the master and sub-column generation product targeting problems using C++ and Cplex,

4. Providing a detailed mathematical derivation of the dual master problem related to the product targeting problem as well as the reduced cost function required to compute the sub-problem objective function.

5. Introduction of parallel processing coding technology within the sub-problem formulation of the product targeting problem to speed up solution time,

6. Incorporation of dual smoothing variables in order to reduce the dual variable heading-in, yo-yo, and tailing-off effects,

7. Computation of upper and lower bounds to calculate the optimality and integrality gaps for the complex product targeting problem as the model progresses through the iterative steps,

8. Developed a methodology to remove unused columns as superior columns enter the column generation algorithm. This method does, however, sacrifice accuracy in order to allow loosely constrained product targeting problems to get closer to a reasonable answer within a shorter time frame,

9. Utilising algorithmic improvements in order to solve significantly larger product targeting problems by reducing memory consumption and computational overhead (i.e., limited columns being generated),

10. Coding preceding algorithms from first principles instead of calling predefined libraries. This allows for easy manipulation and adaptation of the algorithms when required.

After summarising the contributions made throughout this thesis, the tangible model results provided in Chapters 7, 8, and 9 are considered. In these chapters, the capability of these newly proposed methodologies in solving more complex and constrained product targeting optimisation problems within a reasonable time frame is practically proven. Furthermore, the ability of the algorithms to reduce computational memory requirements is also demonstrated.

Chapter 7

Input data and verification testing

The following chapters present the computational results obtained from the product targeting optimisation problem formulations, as outlined in Chapters 4 and 6. Technical evaluation techniques such as verification and validation testing are applied to assess the validity of the models under consideration. In this chapter, model verification is conducted to evaluate model accuracy and to establish if the model's response is as expected. The verification process also entails critically evaluating the programming code used to implement the models to assess the correctness thereof. The validation process is concerned with the ability of the optimisation models to solve realistically sized problems with a specific focus on solution time, memory consumption, and model accuracy. However, the results generated by the validation testing will only be discussed in Chapter 8, while the verification testing outcomes will be presented in Chapter 7. In many cases, the verification and validation processes are linked to one another, therefore both these evaluation techniques are addressed in this thesis.

The verification process is performed by comparing the IP formulation model outcomes of 3 predefined test cases with the column generation model for the same test cases. This verification process aims to determine if both modeling methodologies obtain the same objective function values at the termination point of the algorithms. A comparison is also performed on the optimality gap obtained, the computational time required, and the memory consumed for the various verification test cases. The verification test cases do not exceed 3000 customers, 15 products, and 3 channels in an attempt to try and simplify the process of critically evaluating and comparing the various models with one another. Using extensive test cases to perform model verification will unnecessarily complicate the interpretation of the model results, which could lead to incorrect conclusions being made.

The product targeting problem instances are solved using the commercial solver Cplex coupled with C++ code designed within the integrated development environment (IDE) known as Visual studio. The user-defined input data is introduced into the optimisation models by linking the C++ code to Microsoft Excel files containing fictitious data generated using Jupyter Notebooks as IDE and Python as coding language. A detailed explanation will be provided in Section 7.3 on the inner workings of the data acquisition and generation process. The solution algorithms implemented within Cplex to solve the various optimisation models include the branch-and-bound algorithm when considering the IP formulation method, whereas, for the column generation method, both branch-and-bound, as well as the simplex method, are leveraged. The input data was generated randomly to simulate the different product targeting scenarios. Still, an effort was made to ensure that the data is still a realistic representation of reality, even though it is fictitious. The specifications of the computer used to solve the various test cases and generate the results reported in Chapters 7 and 8 are as follows: Processor Intel(R) Core(TM) i7-7500U CPU @ 2.70 GHz 2.90 GHz, Installed RAM of 8 GB and a 64-bit operating system leveraging an x64-based processor. Note, however, that the code was generated using a virtual box environment with Linux as the operating system. The virtual environment only had 4 GB of RAM for usage during computational processes.

7.1 Setting the scene

A combination of customers, products, and channels should be considered to generate results for the verification and validation tests. For this thesis, the number of customers and products considered across the various test cases is dynamic, increasing or decreasing depending on the test case under consideration. However, the number of channels is kept constant at a value of 3 (voice, SMS, and email), with no additional channels being considered. Adding additional channels would result in the entire mathematical model requiring refactoring, which is outside the research scope and will therefore be proposed as a potential future research avenue to explore. It is also assumed that the input data required for the optimisation models, generally obtained from propensity/machine learning models, has already been developed and is available for use within the optimisation framework. In reality, the end user would first need to develop the relevant base models before designing and implementing the optimisation models because, without the base data, the optimisation models will not be able to serve their purpose. Refer to Chapter 2 for details regarding the setup of the relevant propensity and ML models. In the subsequent section, examples are provided on the type of data points required as input data into the optimisation models to compute the results displayed in Chapters 7 and 8.

7.2 Model data

The input data utilised within the verification and validation test cases are structured similarly to the data in Tables 7.1 – 7.9. The difference, however, is that the size of data instances (i.e., number of customers and products) would vary depending on the test cases being considered. Therefore, the data in the tables below are only an extract of the data used within Chapters 7 and 8 to generate the required model results. In addition, the mentioned data points are generated using the Jupyter IDE code set, specifically developed for the various optimisation algorithms. More detail regarding the Jupyter IDE will be provided in Section 7.3.

Customer i	Product j	Channel u	Cost $c_{iju}^{(v)}$ (R)	Profit p_{iju} (R)	Probability Reach r_{iju}	Channel Preference $c_{iju}^{(p)}$	Cross Sell Option $c_{iju}^{(s)}$
1	1	1	35	75	0.35	0	0
1	1	2	15	105	0.85	1	0
1	1	3	75	163	0.25	0	1
1	2	1	65	85	0.12	0	0
1	2	2	43	210	0.35	0	1
1	2	3	55	63	0.95	0	0
1	3	1	12	67	0.04	0	0
1	3	2	97	183	0.71	0	0
1	3	3	73	96	0.76	1	0
2	1	1	14	175	0.32	0	0
2	1	2	75	425	0.65	0	1
2	1	3	35	43	0.89	0	0
2	2	1	31	65	0.52	0	0
2	2	2	72	116	0.85	0	1
2	2	3	65	94	0.72	0	0
2	3	1	72	157	0.66	1	0
2	3	2	97	183	0.31	0	0
2	3	3	39	78	0.11	0	0

Table 7.1: Fictitious input data example: customer, product and channel related parameters

The data presented in columns 1 to 3 of Table 7.1 refers to the number of customers (2), products (3), and channels (3) being considered for this example. The preceding columns are utilised as indexing parameters to feed the relevant input data points into the correct indices of the optimisation algorithm variables. Column 4 refers to the rand value cost associated with offering product j to

a given customer i. In contrast, column 5 corresponds to the rand value income obtained from the same offering. When offering a product j to a customer i via a selected communication medium u, there would exist a probability of reaching customer i for said product j on the specific channel u. The probability information is given in column 6 of Table 7.1. The preceding information is used to augment the profitability of each customer within the model objective functions, as the ability to reach a customer should also influence the model's decision about the customer. Column 7 in Table 7.1 refers to data regarding channel preference, whereas column 8 provides information regarding cross-selling availability linked to each product being offered via a selected channel.

In Table 7.2, the first column refers to the product indexing parameter used within the optimisation algorithm variables. In column 2, the fixed cost associated with offering the product j to customers is presented. When offering product j to customers, it is also imperative to guide the model on the number of minimum and maximum customers allowed to receive the specific product offering. The maximum number of customers allowed to receive the product offering is reported in column 3 of Table 7.2, with the minimum customer requirements being provided in column 4 of the same table. The last column within Table 7.2, column 5, contains data regarding the budgeting parameter linked to each product j. Given that there are costs associated with offering product j to customers, the model is required to govern the number of offerings per product j in such a manner as to prevent the offering cost from exceeding the annual budgeting requirements.

Product j	Fixed Cost $c_j^{(f)}$ (R)	Max Customers $l_u^{(j)}$	Min Customers $l_u^{(l)}$	Budget B_j (R)
1	30	100	35	15497
2	74	285	82	19364
3	24	47	22	4924

Table 7.2: Fictitious input data example: product related parameters

Table 7.3 contains data regarding channel decisioning parameters with column 1 representing the channel indexing parameter used within the optimisation algorithm variables. When leveraging specific channels to position product offerings to customers, it is required that the model adheres to certain operating bounds, such as the maximum and minimum products allowed to be offered via a specific channel u. Column 2 provides examples of the maximum number of products allowed to be offered via a specific channel u whereas column 3 represents the minimum product requirements for said channel.

Channel u	Max Products $c_u^{(m)}$	Min Products $c_u^{(l)}$
1	70	27
2	74	42
3	94	20

Table 7.3: Fictitious input data example: channel related parameters

In the following table, Table 7.4, data regarding product and channel constraints are reported. The product and channel indexing parameters are provided as used in the optimisation algorithm variables in columns, 1 and 2. In column 3, information is portrayed regarding the exclusion of specific channels for a given product j. The preceding could coincide with product houses having certain prerequisites for which channels should be leveraged to position their products. The data in columns 4 and 5 are similar to that in Table 7.3. However, the data in Table 7.4 provides minimum and maximum requirements per product and channel level, whereas Table 7.3 only focuses on a channel level. With that said, column 4 provides information regarding the maximum number of products j which can be offered to customers via a selected channel u. Similar is true for column 5, with this column guiding the minimum number of products j allowed to be offered using channel u.

Product j	Channel u	Exclude Channel E_{ju}	Max Products Per Channel $l_{ju}^{(m)}$	Min Products Per Channel $l_{ju}^{(n)}$
1	1	1	10	5
1	2	0	23	11
1	3	0	43	28
2	1	0	47	3
2	2	1	73	4
2	3	0	32	18
3	1	0	15	1
3	2	0	42	7
3	3	1	79	31

Table 7.4: Fictitious input data example: product and channel related parameters

The first four columns of Table 7.5 is related to the indexing parameters used by the optimisation algorithm variables to feed input data into the models. Column 5 in Table 7.5 provides information regarding the probability of obtaining a right party contact (RPC) when trying to contact a given customer i via a specific channel u in an attempt to position product j. However, the probability of reaching the intended customer depends on many factors, such as the quantity of phone numbers or email addresses available for that given customer, the type of product being offered, and potentially even the time of day the attempt was made. Therefore, one requires the optimisation model to select the most probable RPC number or email address to increase the chances of reaching the intended customer. The RPC probability is used in conjunction with the answer probability in Table 7.1 to guide the optimisation model to decide which number or email option it should use to maximise the chances of reaching the required customer i for a given product j through channel u. The last column within Table 7.5, column 6 provides the optimisation model with information regarding marketing consent obtained for positioning offer j to customer i via channel u. Suppose the marketing consent is reported as a value 1. In that case, it is indicative that the associated product j may not be offered to customer i via the channel u under consideration. The contrary is also true when marketing consent is assigned a value of 0. In this case, the financial institution is allowed to contact customer i via channel u in order to promote product j.

Customer i	Product j	Channel u	Options Per Channel q	Probability RPC $c_{ijuq}^{(q)}$	Marketing Consent $c_{ijuq}^{(pr)}$
1	1	1	1	0.5	1
1	1	2	1	0.11	0
1	1	3	1	0.28	0
1	1	3	2	0.78	0
1	2	1	1	0.43	1
1	2	2	1	0.65	0
1	2	3	1	0.12	0
1	3	1	1	0.93	1
1	3	2	1	0.65	0
1	3	2	1	0.35	0
1	3	3	1	0.42	0

Table 7.5: Fictitious input data example: customer, product, channel and options per channel

Table 7.6 provides information regarding the contact recency of customers. When a financial institution has recently contacted a customer in an attempt to sell a product, for example, there are some pressure rules applied (this might vary per organisation and per type of offer being positioned), which prevents the optimisation model from including the affected customer into the decision process until a certain time has lapsed. In such a case, the recency parameter for the affected customer will

be assigned a value of 1. However, the opposite is also true. If a customer is allowed to form part of the model decision process and he/she does not have any recency exclusion triggers, the recency parameter will take on a value of 0.

Customer i	Recency $r_i^{(c)}$
1	0
2	0
3	1

Table 7.6: Fictitious input data example: customer related parameters

In Table 7.7, the optimisation model is provided with customer input data specifying if a customer i can contact the financial institution via channel u. Suppose column 3 is assigned a value of 0 for the channel exclusion parameter mentioned. In that case, it is indicative that customer i has not provided any instruction to exclude channel u as the contact medium. Therefore the affected channel could be utilised within the model decisioning framework for said customer. However, suppose a value of 1 was assigned. In that case, the affected channel should be excluded from the decision framework, and the model would need to select an alternative medium of contact for customer i.

Customer i	Channel u	Exclude Channel $c_{iu}^{(e)}$
1	1	0
1	2	0
1	3	0
2	1	0
2	2	1
2	3	0
3	1	0
3	2	0
3	3	1

Table 7.7: Fictitious input data example: customer and channel related parameters

The data in Table 7.8 is related to the cross-sell parameters for the various optimisation models developed within this thesis. The first 3 columns represent the indexing parameters used by the optimisation algorithm variables to feed input data into the various models. Column 4 refers to the profit which could be made by the financial institution when offering product j to customer i, but then afterward cross-selling product c to the same customer. In effect, the preceding represents selling a customer an additional product after success with the first sell attempt. There might be multiple cross-selling options c associated with a single product j, as seen in Table 7.8. Therefore the optimisation algorithm would need to consider all available options and limitations before proposing the best cross-selling option for each given customer (if available as an option).

Customer i	Product j	Cross Sell Products c	Cross Sell Profit $c_{ijc}^{(r)}$
1	1	1	53
1	1	2	13
1	1	3	85
1	4	1	83
1	4	2	51
2	3	1	24
2	3	2	72
2	3	3	31

Table 7.8: Fictitious input data example: customer, product and cross-sell related parameters

The last table contains data on the time of day parameters considered within the optimisation models to determine the best time to call a customer if voice has been selected as the medium of contact. Note that the optimisation models are only set up to account for the morning, afternoon, and evening periods. Therefore the model would only be able to select from a list of three time periods for each cell phone number q linked to a customer i. The data used to guide the selection of the best time t to call a customer i is seen in column 5 of Table 7.9. In this column, the model is provided with the probability of an answer for a given customer i across the various periods t. The higher the probability, the better chances the end user will have to get hold of the affected customer when using the selected number q during the identified period t. The first 4 columns in Table 7.9 pertain to the indexing parameters used by the optimisation algorithm variables to feed the below input data into the various optimisation models.

Customer i	Channel u	Options Per Channel q	Time of Day t	Probability Answer $c_{iuqt}^{(t)}$
1	1	1	1	0.53
1	1	1	2	0.97
1	1	1	3	0.21
1	1	2	1	0.43
1	1	2	2	0.27
1	1	2	3	0.71
2	1	1	1	0.63
2	1	1	2	0.31
2	1	1	3	0.17
3	1	1	1	0.93
3	1	1	2	0.63
3	1	1	3	0.53

Table 7.9: Fictitious input data example: customer, product and cross-sell related parameters

Note that the above tables are utilised as examples to give the reader insights into the input data for the product targeting optimisation problem. As previously mentioned, the amount of data required will depend on the problem instance's size and complexity to be solved. In Section 5.3, a discussion is provided on the scalability of the various optimisation models when increasing the data feed into the models.

7.3 Generation of fictitious model data

The fictitious data utilised as input to the product targeting optimisation models discussed throughout Chapters 7 and 8 (test instances 1 – 17) were generated using python code. The mentioned code was developed within an integrated development environment known as Jupyter Notebooks.

Within the python coding framework, the end user must provide the environment with the number of customers, products, and channels required to generate the problem instance. In addition, the end user should also provide the operating regime (minimum and maximum ranges) related to the number of customers, products, and campaign budgets as input to the python code required for the data generation process. After receiving the preceding inputs, the python code runs through many for-loops designed to construct the data into the required format, as deliberated in Section 7.2. Finally, the output of the data generation process is saved as a .xlsx file compatible with the c++ code used to construct both the IP formulation and the column generation models. Figure 7.1 provides a view of the Jupyter Notebook Python file.

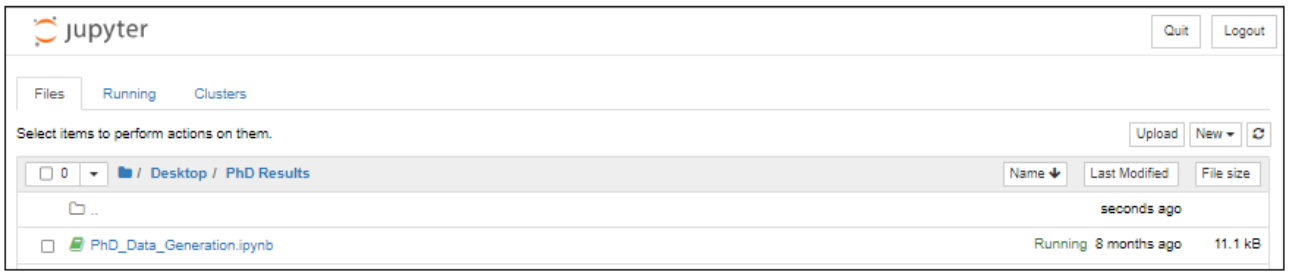

Figure 7.1: Input data generation

In order to ingest the data contained within the .xlsx file, an excel reader is utilised within the c++ coding environment. The methodology used for data generation and transformation generated files containing less than a million rows which warranted the use of excel as a data handling tool. However, once the end user wants to solve optimisation problems containing datasets larger than 1 million rows, excel as a technology would need to be replaced using technologies such as SQL or Postgres databases.

7.4 Problem scalability

As noted in the literature study provided in Chapter 3, it is apparent that the product targeting problem is an NP-hard problem. This means that by increasing the problem size, the processing time and computational requirements increase exponentially to a point where a given server cannot provide enough resources to solve the problem under consideration. Considering the preceding, it is already known that the problem is NP-hard and that by increasing the complexity of the problem as was done in this research study, it will remain NP-hard, and therefore no NP-hard test was performed. However, to compare the performance of the proposed IP formulation and column generation algorithms, a test was constructed to evaluate the solution time of each algorithm when scaling the input data. The results of these tests are provided in Figures 7.2 and 7.3, respectively.

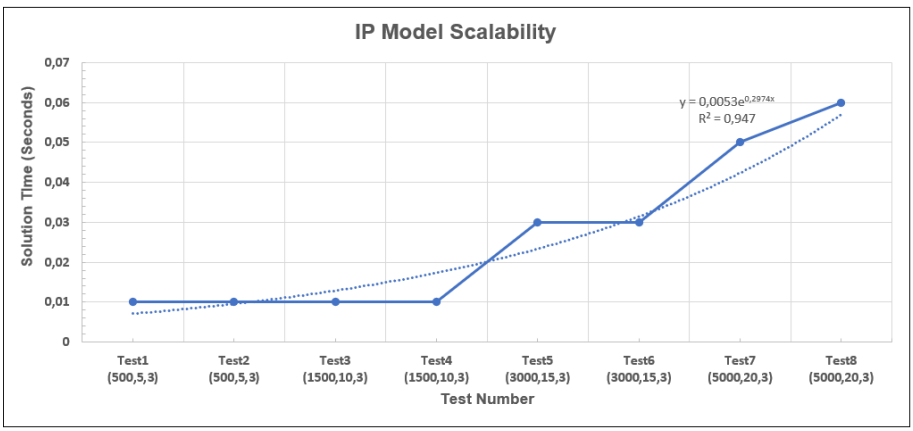

Figure 7.2: IP formulation scaling analysis

Figure 7.2 provides the computational time required by the branch-and-bound algorithm to solve the IP formulation test cases 1 to 8. The values portrayed on the x-axis labeling are structured according to (Number of customers, Number of Products, Number of channels) to provide insight on the size of the problem instances solved during each text case. Note that test case 1 starts with a relatively small optimisation problem where the problem size is gradually increased up to the point of test case 8. The y-axis provides some insights into the time taken to solve each of the mentioned problem instances. The branch-and-bound algorithm is quite effective in solving these mentioned problem instances within a few seconds. However, as the size of the problem instances increases, an exponential trend in solution time is observed, even though it is quite fast and efficient. While applying the branch-and-bound algorithm to the various IP formulation test cases, it was noted that the limiting factor for finding a global optimal solution for the branch-and-bound algorithm was not necessarily solution time but rather memory requirements. When the algorithm has enough memory assigned to it, the branch-and-bound method could quickly solve the problem. The problem is, however, that one does not have an unlimited amount of memory to assign to the problem. In the subsequent chapters, an elaborate discussion will be provided on these mentioned limitations. For this reason, the column generation algorithm was also explored in this research study, with the results for this algorithm being depicted in Figure 7.3.

Figure 7.3 depicts the computational time required by the column generation algorithm to solve test cases 1 to 15. When evaluating the y-axis of Figure 7.3, it is apparent that the column generation algorithm takes longer to solve the various test cases than the IP formulation. Although both algorithms can reach the global optimal solution, the column generation algorithm has much more overhead to manage in computing solutions. On the other hand, the column generation algorithm can solve much larger problem instances without running into memory limitations. More detail will be provided on the preceding in Section 7.5.

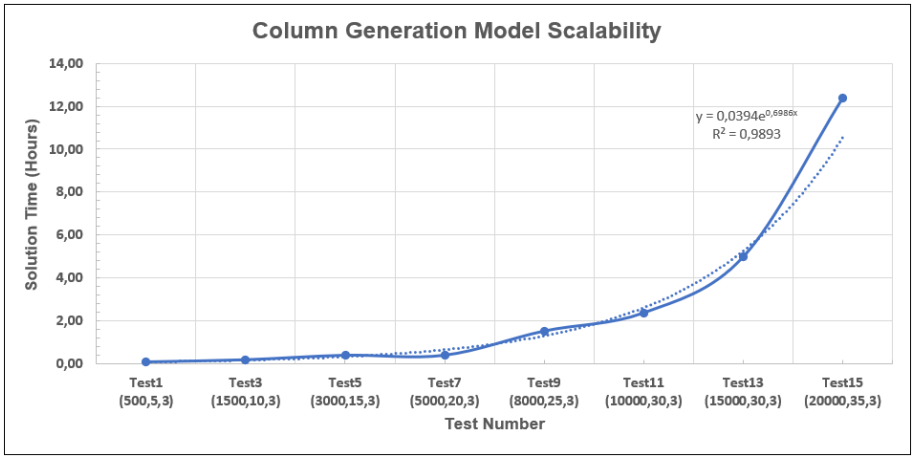

Figure 7.3: Column generation scaling analysis

The above information provides the reader with insights into the algorithmic responses that could be expected when applying both the IP formulation and column generation algorithms to scaled problems. A more detailed comparison between the two methodologies is provided in the subsequent sections, but a keynote to take from the above data is that when scaling problem instances related to product targeting use cases, it will result in the problems becoming more complex to solve as well as require the various algorithms to leverage more compute power to solve said instances. In such cases, some algorithms can only solve the mentioned problems to a point where the algorithms might start running out of memory or require extensive computational time rendering the algorithm unpractical to implement within real-life scenarios. Information regarding the various test cases as depicted in Figures 7.2 and 7.3 (i.e., number of customers, product, and channels) are provided throughout the subsequent sections of Chapters 7 and 8.

7.5 Novel IP formulation and column generation verification

The verification tests for the IP formulation and column generation algorithms were performed using 3 different test cases. The first test case (Test 1) consists of 500 customers, 5 products, and 3 channels which were being considered within the optimisation framework. The second test case being utilised for verification testing (Test 3) incorporated 1500 customers, 10 products, and 3 channels as part of the problem instance. Finally, the last test case (Test 5) was scaled to a problem size containing 3000 customers, 15 products, and 3 channels. The three preceding test cases were tightly constrained, allowing the optimisation models to compute global optimal solutions quite easily. Tightly constrained in this context refers to the upper and lower bound constraints of the product targeting problems being limited in such a way as to provide the optimisation algorithms with limited options to choose from. In turn, it allowed the algorithms to reach global optimality within limited compute time while consuming a fraction of the memory compared to realistically sized problems. A tabulated format of the test case parameters under discussion is provided in Table 7.10 below. Column 2 in Table 7.10 refers to the algorithms applied to the given test cases, with term IP referring to the IP formulation algorithm. In contrast, the term DM refers to the decomposed column generation algorithm. The same acronyms will be utilised throughout the following tables to refer to the relevant optimisation algorithms.

Test Case	Model Type	Num Cust	Num Prod	Num Chan	Type Constrained
Test1	IP/DM	500	5	3	TIGHT
Test3	IP/DM	1500	10	3	TIGHT
Test5	IP/DM	3000	15	3	TIGHT

Table 7.10: IP formulation and column generation model verification inputs (tight constrained)

Throughout the verification section, the focus was primarily on evaluating the correctness of the code utilised to construct the IP formulation and column generation algorithms rather than the ability of the mentioned algorithms to solve complex problems. It is for this reason that simplistic test cases were constructed in order to see if both algorithms are capable of reaching global optimality. In addition, the objective function results obtained for each test case after applying the two optimisation algorithms were compared to one another to identify if they could obtain identical objective function values. By proving that both algorithms can select the same combination of customers, products, and channels, which results in the same objective function values being computed, it can be concluded that the mathematical formulation and back-end coding is sound. However, when using complex large scaled optimisation problems for verification testing, it is not guaranteed that the algorithms will be able to reach global optimality, as seen in Chapter 8. Therefore there will not be any way to sanity check and compare the two algorithms with each other. This is why verification testing was done on oversimplified test cases. Insights into the model results obtained for the preceding test cases are provided in Sections 7.5.1 and 7.5.2, respectively.

7.5.1 Novel IP formulation model verification results

The IP formulation algorithm solved Test 1, as depicted in Table 7.11, within a few seconds. The objective function value obtained was R 212 448, while the number of columns and rows generated during the solution process corresponded to values of 57894 and 47203, respectively. Note that the memory consumed by the problem instance was around 0.03 GB which is quite marginal when considering optimisation problems. Test 3 was solved by the branch-and-bound algorithm within 0.01 seconds while consuming approximately 0.17 GB of memory. The objective function value was computed at R490 716, whereas the number of columns and rows for this test case was 345425 and 276528, respectively. Note that with an increase in the number of customers and products, the optimisation model is required to generate more columns and rows due to the increased number of combinations added to the problem. Although the size of the problem is increased, the IP formulation handles it effectively. The last verification test case, Test 5, generated 1042172 columns and 826313 rows with a calculated objective function of R524 748. For Test 5, there was a slight increase in the

computational time, as depicted in Table 7.11, with a value of 0.03 seconds being reported. The memory consumed by Test 5 was seen to be 0.53 GB which is quite an increase from the previous test case. From the above test cases, it is apparent that when scaling the size of the problem to be solved, the solution time and memory requirements will increase when using the IP formulation algorithm. It is imperative to note that each of the three test cases being considered obtained an optimality gap of 0%, indicating that the IP formulation achieved global optimality in each scenario. The downside of the IP formulation is that the number of columns and rows increases exponentially as the problem instance is scaled, resulting from the algorithm requiring all of the available decision combinations in memory. A detailed view of the memory consumed over time for each test case is depicted in Figure 7.4.

Test Case	Objective (R)	OPT GAP (%)	Time (sec)	Num Cols	Num Rows	Memory Consumed (GB)
Test1	212448	0	0,01	57894	47203	0,03
Test3	490716	0	0,01	345425	276528	0,17
Test5	524748	0	0,03	1042172	826313	0,53

Table 7.11: IP formulation model verification output (tight constrained)

The top image within Figure 7.4 depicts the memory consumption recorded for Test 1. No significant step change was noted in the actual and the swap memory for the test case under consideration. The red block indicates at what point the solution algorithm was deployed to generate the global optimal results, as noted in Table 7.11. The middle image is representative of the memory over time response documented for Test 3. In this instance, the reader can already start to notice a small step change in the memory consumed while computing the optimal solution. It is, however, insignificant, as the computer still has capacity on both the original and the swap memory. A noteworthy step change was recorded in the memory requirements for Test 5. In this instance, the original memory increased rapidly from the 50% to 75% memory mark within a few seconds. There was still much free capacity left on the computer. However, a noticeable change in the compute capacity required for the IP formulation algorithm was reported as the problem instance was being scaled. An important aspect to note is the solution time reported in Table 7.11 compared to the time in seconds seen in Figure 5.4. Note that the time under consideration in Figure 5.4 is longer than that reported in Table 7.11. The delta difference could be attributed to the time taken to load all of the data into the various arrays and structure it in such a manner as to allow for consumption within the various optimisaton models. In test cases where the solution time is much longer than the display time, as noted in the memory and swap history console, only a snippet is provided to supply the reader with proof of the memory utilisation for the various problem instances. Such examples exist for the verification tests performed within Section 7.5.2. The preceding was, however, fine within the verification test instances seen for the IP formulation due to the short solution time utilised to compute an answer.

It is imperative to note that there is a difference between the memory consumption reported in Table 7.11 and that displayed in Figure 7.4. The memory consumption reported in Table 7.11 portrays the memory requirements of the optimisation algorithm in isolation whereas the memory consumption depicted in Figure 7.4 is indicative of the memory required to run the virtual environment, process the input data, load the data into the relevant arrays and execute the optimisation algorithm. That is why there is a discrepancy when evaluating the reported memory consumption for Test 1, for example. The model memory consumption is reported as 0.03 GB in Table 7.11, while the total process memory consumption is seen to be 2.2 GB in Figure 7.4. This delta memory difference interpretation needs to be considered throughout the verification and validation testing phases, as it is applicable across the board for all test instances.

Figure 7.4: IP Formulation verification tests memory consumption

In table 7.12, a view is provided to the user regarding the solution status of each test case attempted within Section 7.5.1. When a "success" status is assigned to the test case, the solution algorithm can solve the optimisation problem without encountering issues such as infeasible solutions, excessive compute time, or out-of-memory limitations. On the other hand, if an algorithm encounters the preceding, a "failed" status will be assigned to the affected test case. When viewing Table 7.12, it can be concluded that the branch-and-bound algorithm leveraged for the IP formulation successfully solved the three test cases attempted within the verification section of the research study.

Test Case	Status
Test1	Success
Test3	Success
Test5	Success

Table 7.12: IP formulation verification end state

In the subsequent section, the column generation algorithm's capability to solve the same test cases to global optimality, as was done in the above section, is evaluated. Information is provided regarding the solution outcomes, with the results being compared to those reported in Section 7.5.1. The outcome of the comparative study will be utilised to prove the validity of both algorithms, with the results forming the baseline for the validation tests that will be discussed in Chapter 8.

7.5.2 Novel column generation verification results

The results obtained for verification test cases 1, 3, and 5 are stipulated in Table 7.13. Table 7.13 contains more information when compared to the solution parameters seen in Table 7.11. The preceding includes aspects such as the number of iterations executed by the column generation algorithm, the upper and lower bound values, and the computation of the integrality gap. The mentioned parameters are, however, only relevant to the column generation algorithm and are not of interest when generating results using the IP formulation (branch-and-bound algorithm).

When evaluating the results obtained for Test 1, it is apparent that the column generation algorithm could obtain the same global optimal solution similar to what was recorded in Table 7.11

for the IP formulation. Both algorithms computed an objective function value of R212448, indicating that the back-end coding is sound for both algorithms. It is, however, imperative to note that the column generation algorithm required a significantly longer time to compute the same global optimal solution. A solution time of 0.08 hours for Test 1 was recorded when applying the column generation algorithm compared to the IP algorithm, which computed an answer within 0.01 seconds. It is clear that when looking at the computational time, the IP algorithm trumps the column generation algorithm completely. The reason an increase in the solution time is seen is a result of the computational overhead that needs to be managed by the column generation algorithm. An upside to using the column generation algorithm is identified when analysing the memory requirements for the two algorithms used to solve Test 1. It is noted that the memory consumed for the column generation algorithm equated to a value of 0.01 GB. In contrast, the IP algorithm required 0.03 GB of memory to derive the same solution. The difference in memory requirements for Test 1 does not seem noteworthy at this stage. However, the gain in memory reduction will only be fully appreciated when scaling the problem instances as seen throughout the later test cases (Test 3 – Test 12). The reduction in memory requirements for the column generation algorithm can be attributed to the reduced number of columns and rows considered to compute the solution. For Test 1, a total of 755 columns and 2043 rows were generated.

When comparing that to the test results in Table 7.11 for Test 1, it can be concluded that the number of columns and rows generated by the column generation algorithm is drastically lower. A total of 150 iterations were required to reach the global optimal solution, as deliberated in Table 7.13 for Test 1. Note that the upper and lower bound is utilised in the column generation formulation to compute the integrality and optimality gaps, as explained in Chapter 6. The integrality gap calculated for Test 1 equated to a value of 1. The optimality gap calculated for Test 1 was reported as 9.82%. Even though the optimal solution has been obtained (i.e., confirmed using the objective function value), the column generation algorithm must run through multiple iterations to discount possible extreme points to reduce the optimality gap to 0%. The algorithm was therefore terminated at less than 10% to limit the computational time.

A similar trend was noted when evaluating the computational results obtained for Test 3. Both the column generation and IP formulation algorithms were able to compute an objective function value of R490716, confirming the sanity of the coding logic utilised. The solution time consumed to compute the global optimal solution for Test 3 equated to a value of 0.18 hours which was much longer than the time required for the IP formulation to find a solution. The memory requirements for the column generation algorithm were reported at a value of 0.01 GB, significantly less than the 0.17 GB required by the IP formulation. The column generation algorithm performed a total of 94 iterations to reach a point of global optimality. The integrality gap for Test 3 was computed at a value of 1, whereas the optimality gap equated to a value of 5.85%. When applying the column generation algorithm to Test 3, a limited number of columns (950) and rows (6078) were computed, remarkably less when compared to the results reported in Table 7.11 for the IP formulation.

The last problem instance being considered for verifying the validity and soundness of the developed algorithms is Test 5. The results obtained for Test 5 are reported in Table 7.13 after applying the column generation algorithm to solve the problem instance. The column generation algorithm reached the point of global optimality corresponding to an objective function value of R524748, identical to the results noted in Table 7.11 for the IP formulation. The solution time of 0.39 hours was significantly longer than the 0.03 seconds reported for the IP formulation. The memory consumed during the computational process was only a marginal 0.02 GB, which drastically improved from the sizeable 0.53 GB required by the IP formulation. Again, the integrality gap was calculated at a value of 1, whereas the optimality gap equated to a value of 4.67%.

Test Case	Iter	MSTR OBJ (R)	Upper Limit (R)	Lower Limit (R)	INT GAP	OPT GAP (%)	Time (hrs)	Num Cols	Num Rows	Mem Cons (GB)
Test1	150	212448	240935	212448	1	9,82	0,08	755	2043	0,01
Test3	94	490716	521211	490716	1	5,85	0,18	950	6078	0,01
Test5	76	524748	550441	524748	1	4,67	0,39	1155	12113	0,02

Table 7.13: Column generation verification output (tight constrained)

A similar memory consumption analysis is provided for the column generation verification test results in Figure 7.5, as to what was presented in Figure 7.4 for the IP verification tests. The top image in Figure 7.5 depicts the memory usage during the execution of Test 1. In this instance, no significant increase in memory usage is noted for both the actual or the swap memory regions. In addition, contrary to what was discussed in Section 7.5.1 for Tests 3 and 5, no significant increase in memory consumption is noted for the same test cases reported in Section 7.5.2.

Figure 7.5: Column generation verification tests memory consumption

By applying the column generation formulation, memory requirements were reduced by only considering a subset of the columns in the product targeting problem formulation. As a result of the preceding, no step change in memory consumption is noted for Tests 3 and 5 in Figure 7.5 for both the actual and the swap memory regions. Note that the processing time emphasised in the red blocks of Figure 7.5 is only snippets at the end of the computational process and does not represent the time taken to solve the various test instances. The memory consumption snippets were explicitly taken at the end of the computational process to ensure that it is representative of the memory consumed at the point when the column generation algorithm obtained the global optimal solution.

The results from Table 7.14 demonstrates the capability of the column generation algorithm in solving the various test cases reported in Table 7.10. The column generation algorithm solved Tests 1, 3, and 5 to global optimality without encountering computational issues such as memory limitations or infeasible regions. In each of the mentioned test cases, the column generation algorithm improved the memory consumed compared to the IP verification tests while significantly increasing the solving time required due to the computational overhead required by the solution algorithm. In addition, none of the test cases under consideration terminated in a "failed" end state.

Test Case	Status
Test1	Success
Test3	Success
Test5	Success

Table 7.14: Column generation verification end state

Considering the verification data for the IP and column generation algorithms, they could solve the various test cases within a reasonable time frame without any computational limitations. The above results also verify the validity and soundness of the algorithms being utilised as both methodologies computed the same global optimal solutions for each test case under consideration. The preceding outcomes also verify the correctness of the code implemented, ensuring that no bugs or incorrect mathematical formulations are being leveraged throughout the various algorithms. From the preceding results, it can also be derived that the column generation algorithm is significantly more efficient in managing the memory required to solve the various optimisation instances. The column generation algorithm will also solve significantly larger product-targeting problem instances compared to the IP formulation due to the improved memory management ability. The statement above will be substantiated when performing the validation testing, as noted in Chapter 8. The only downside which could be identified throughout the verification tests, when critically evaluating the column generation algorithm, is the increased computational time required to compute the global optimal solution. However, the algorithmic improvements and memory management capabilities of the column generation algorithm outweigh the negatives by a notable margin. An in-depth discussion regarding the various algorithm capabilities and limitations is presented in Chapter 8.

Chapter 8

Validation testing

The validation process is performed by comparing the IP formulation model performance to that of the column generation model when applied to a multitude of test cases reaching up to 30000 customers, 35 products, and 3 channels. These test cases are structured so that it evaluates the various models' solution time, memory consumption, the number of columns and rows generated, and the optimality gap obtained for each modeling technique. The preceding will prove the superiority of the column generation algorithm in handling scaled product targeting optimisation problems compared to the IP formulation modeling framework.

Validation testing performed on the IP formulation consisted of 9 test instances, while 20 problem instances were used to validate the computational capabilities of the column generation algorithm. Fewer problem instances were utilised to test the capability of the IP formulation due to the limitations experienced by the branch-and-bound algorithm in solving problem instances consisting of more than 5000 customers, 20 products, and 3 channels. Details regarding the IP formulation limitations are provided in Section 8.1. The initial test instances used for validation purposes on both optimisation algorithms (IP and column generation) are provided in Table 8.1. Test 7 was the first validation test instance consisting of 5000 customers, 20 products, and 3 channels. The algorithms were also applied in solving test instances Test 9 (which comprised 8000 customers, 25 products, and 3 channels) and Test 11 (composed of 10000 customers, 30 products, and 3 channels), respectively, to test the capability of the proposed formulations in solving larger problem instances. During validation testing, it was identified that the optimisation algorithms responded differently to constraint bounds variations; therefore, tests were constructed using loose and tight formulated constraints. Problem instances Test 7, 9, and 11 were tightly formulated, constraining the upper and lower bounds in such a manner as to prevent the optimisation models from having a multitude of possibilities to choose from. Details regarding the loosely constrained test instances will be provided throughout the subsequent paragraphs.

Test Case	Model Type	Num Cust	Num Prod	Num Chan	Type Constrained
Test7	IP/DM	5000	20	3	TIGHT
Test9	IP/DM	8000	25	3	TIGHT
Test11	IP/DM	10000	30	3	TIGHT

Table 8.1: IP and column generation solution validation input (tight constrained)

To push the limits of the column generation algorithm, larger tightly constrained test cases were formulated to investigate the scalability of the column generation algorithm's solution capabilities. Information regarding the mentioned instances is denoted in Table 8.2. The IP formulation was not applied to the test cases in Table 8.2 as this algorithm reached its computational limits while solving Test 9 in Table 8.1. The initial problem instance in Table 8.2, Test 13, comprises 15000 customers, 30 products, and 3 channels. Compared to Test 9, where the IP formulation reached its computational limits, this instance already contains almost double the number of customers and five

additional products. Therefore, the scale of validation testing for the column generation algorithm was increased even further, with Test 15 comprising 20000 customers, 35 products, and 3 channels. In comparison, Test 16 was increased to a problem size considering 25000 customers, 35 products, and three channels. The final tightly constrained product targeting problem used for validating the column generation algorithm's computational capability was Test 17, with this instance employing 30000 customers, 35 products, and 3 channels. It is noteworthy that the test instances applied to the column generation algorithm were almost three times the size of the problems considered for the IP formulation due to the superior solving capability of the column generation algorithm. The validation test outcomes for the tightly constrained problem instances in Tables 8.1 and 8.2 are provided in Section 8.1 for the IP formulation and Section 8.2 for the column generation formulation, respectively.

Test Case	Model Type	Num Cust	Num Prod	Num Chan	Type Constrained
Test13	DM	15000	30	3	TIGHT
Test15	DM	20000	35	3	TIGHT
Test16	DM	25000	35	3	TIGHT
Test17	DM	30000	35	3	TIGHT

Table 8.2: Column generation validation input

As part of the validation testing performed throughout this thesis, both the IP formulation and column generation methodologies were employed to solve loosely constrained product targeting optimisation problems. Loosely constrained in this context refers to constraints being provided with large upper or small lower bound limits, which results in the optimisation models having a significant search space to traverse to find the global optimal solution. The reason why the mentioned algorithms were also used to solve loosely constrained problem instances and not only tightly constrained test cases, as was considered in Tables 8.1 and 8.2, was to measure the change in the ability of the various algorithms to traverse larger search spaces and how these changes would affect memory consumption, solution time and ability to reach global optimality. The problem instances formulated to test the preceding phenomenon are portrayed in Table 8.3.

The loosely constrained validation testing process is initiated using Test 2, which consists of 500 customers, 5 products, and 3 channels. Note that the baseline parameters, as mentioned for Test 1 in Table 7.10, are identical to Test 2 in Table 8.3, with the only change being the way the various upper and lower bound equations were constrained to generate a loosely constrained mathematical problem. The results obtained for Test 2 after utilising the IP formulation as a solution algorithm are recorded in Table 8.5. To test the flexibility of the column generation algorithm in solving loosely constrained optimisation problems, Test 2 was slit into two problem instances (Test 2_A and Test 2_B), with the results of each shown in Table 8.8. Test 2_A in Table 8.8 refers to the results obtained for Test 2 when utilising the baseline column generation algorithm reported in Section 6.3 to solve the problem instance. In this test scenario, all columns generated by the sub-problem are included in the master problem's computational process to compute a global optimal solution. The methodology discussed in Section 6.3.8 was utilised to compute the results reported in Test 2_B. The preceding was used to test the influence of removing redundant columns from the computational process. The outcomes of the various test cases are discussed in Section 8.2. After solving Test 2, the IP formulation and column generation algorithms were applied to Tests 4, 6, 8, 10, and 12 to evaluate the algorithmic responses when applied to iteratively increasing loosely constrained problem instances. In each of the mentioned test cases, the number of products was expanded to account for 5 additional products per test case. This means that Test 4 accounted for 10 products compared to the 5 products included in Test 2. The same is true for Test 6, consisting of 15 products; Test 8, having 20 products; Test 10, including 25 products; and Test 12, accounting for 30 products. Similar is valid for the number of customers being considered for each one of the test cases. With each iteration, the customer numbers increased, with the final problem instance, Test 12, reaching 10000 customers. During each iteration, the size, complexity, and search space of the

problem instance to be solved is increased significantly to gauge the various algorithms' capability in solving said problems. Note that for Test 14, only the column generation algorithm was used to solve the problem instance, with the IP formulation not being attempted. This is because of the IP formulation's inability to solve Tests 10 and 12, as discussed in Section 8.1. The column generation algorithm was, however, attempted on Test 14 in order to try and compute a global optimal solution to the problem instance under consideration. Test 14 consisted of 15000 customers, 30 products, and 3 channels. When considering the column generation algorithms, each test instance, Test 4, 6, 8, 10, and 12, were split into two test cases (A and B) in order to test the ability of various column generation formulations, as mentioned for Test 2 above, in solving increasingly complex optimisation problems.

Test Case	Model Type	Num Cust	Num Prod	Num Chan	Type Constrained
Test2	IP/DM	500	5	3	LOOSE
Test4	IP/DM	1500	10	3	LOOSE
Test6	IP/DM	3000	15	3	LOOSE
Test8	IP/DM	5000	20	3	LOOSE
Test10	IP/DM	8000	25	3	LOOSE
Test12	IP/DM	10000	30	3	LOOSE
Test14	DM	15000	30	3	LOOSE

Table 8.3: IP and column generation solution validation input (loose constrained)

The results obtained after solving the IP formulation for each loosely defined test case in Table 8.3 are discussed in detail throughout Section 8.1. The outcomes for the column generation algorithm applied to the test cases are reported in Section 8.2. In each of the listed sections, insight is attained regarding the capability of the various algorithms in handling increasingly complex problem instances and which algorithms have superior capabilities in solving said instances.

8.1 Novel IP formulation model validation results

As part of evaluating the outcomes of the tightly constrained problem instances applied to the IP formulation, as seen in Table 8.1, consider Table 8.4. In this table, the results related to Test 7, Test 9, and Test 11 are provided. When assessing the outcomes for Test 7, it is apparent that the IP formulation using the branch-and-bound methodology as algorithm, was able to compute the global optimal solution for the problem under consideration. The algorithmic termination point was reached within 0.05 seconds while the final problem consumed approximately 1.16 GB of memory. The calculated objective function value at the point of global optimality equated to a value of R546618. One can already take note that Test 7 contains quite a significant number of columns (2310938) and rows (1825748), when compared to the test results discussed in Table 7.11, which contributes to the increased memory consumption. More detail regarding memory utilisation will be portrayed in Figure 8.1. When focusing on the results obtained for Test 9 and Test 11, it is apparent that the branch-and-bound algorithm was not able to compute an answer for the product targeting IP formulation. In both instances, the model required more memory than what was available from the server resulting in the termination of the solution algorithm. Therefore, no information could be gathered for Test 9 or Test 11 from a solutioning perspective. One could, however, conclude that the number of columns and rows was significantly more than that generated for Test 7, which in turn requires an increase in compute capacity (i.e. available memory) to derive the global optimal solution. When inadequate memory is available to house the entire problem with all its associated columns and rows within the solution algorithm, the resultant effect will be the termination of the computational process, similar to what was seen in Table 8.4. With early termination, no results could be extracted from the problem instance leaving the end user in the dark.

Test Case	Objective (R)	OPT GAP (%)	Time (sec)	Num Cols	Num Rows	Memory Consumed (GB)
Test7	546618	0	0,05	2310938	1825748	1,16
Test9	Failed	Failed	Failed	Failed	Failed	Failed
Test11	Failed	Failed	Failed	Failed	Failed	Failed

Table 8.4: IP solution validation output (tight constrained)

A detailed analysis of the memory consumed for each test case in Table 8.4 is provided in Figure 8.1. The top image in Figure 8.1 provides insight into the memory utilised during the solution process for Test 7. As noted for this test instance, during the computational process, the entire 4.9 GB of original memory was consumed to compute the global optimal solution. As a result, the algorithm had to start using some of the swap memory (382 MB) to supplement the memory shortfall. Luckily the problem size (number of columns and rows) was small enough that only an additional 18.7% of the swap memory was required to compute a global optimal solution. The above memory response for Test 7 can be noted when referencing the red emphasised block at the top of Figure 8.1. By evaluating the memory requirements for Test 7, it can also be concluded that the IP formulation will not be able to solve significantly larger problem instances than Test 7, as the server only had 1.7 GB of memory available after solving the required instance. Therefore, if there is a need to use the IP formulation to solve larger tightly constrained product targeting problems than Test 7, the available memory on the server would need to be manually increased (capital investment in more RAM for the computer/server) to allow for such computations.

The middle and bottom section of Figure 8.1 provides information regarding the memory utilisation of test cases 9 and 11. From the data depicted in this figure, it is clear that the IP formulation requires more memory than what is available to solve Test 9 and Test 11. In both cases, the original memory has been completely consumed, with the swap memory also being leveraged to progress through the computations. Even though the solution algorithm utilises both original and swap memory, the maximum memory limit is reached, and the process is terminated to prevent a system crash. Note that the image depicted for Test 9 in Figure 8.1 does not reach maximum memory utilisation, but the process did reach its memory limits. The only reason it could not be captured in the image is that the computer froze completely while trying to terminate the code being executed. In Test 11, one can see both original and swap memory reaching its available limits as the computer screen only froze at a point in time where the memory tracker was updated with the actual memory being consumed by the problem instance.

From the evidence provided in Figure 8.1, it is apparent that for the tightly constrained problem instances, the IP formulation can only solve instances up to the size of Test 7 with any problem greater than Test 7 running into memory limitations. The preceding is, however, only applicable to the size of the computer being used to perform the computations (computer specifications provided in the intro to Chapter 7). When increasing the amount of memory available on the computer, larger problem instances could be solved for the IP formulation framework. However, the increase in computing power is outside the scope of this research study and will not be investigated. Instead, identical compute power will be utilised when solving the test instances noted in Table 8.1 using the column generation algorithm to generate a like-for-like performance comparison. This will, in turn, allow for the identification of the pros and cons linked to each algorithm and ultimately conclude on the superior algorithm to use for large-scale product targeting problems. However, before going into too much detail, the focus is first set on the outcomes of the loosely constrained product targeting problems when solved using the IP formulation as solution methodology.

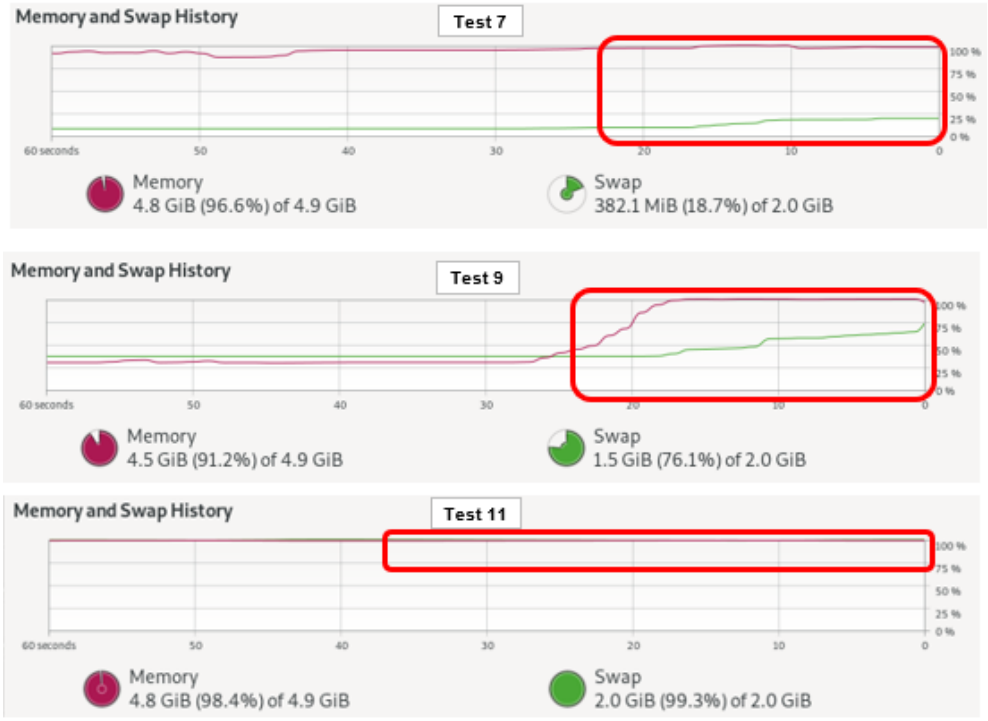

Figure 8.1: IP validation tests memory consumed (tight constrained)

Table 8.5 shows the results generated by the IP formulation when applied to the loosely constrained test cases in Table 8.3. In this table, it can be seen that the branch-and-bound algorithm solved test instances 2, 4, 6, and 8, while problem instances 10 and 12 experienced memory limitations resulting in no solutions being obtained. The memory consumed for Test 2 was reported as 0.03 GB, with the consumption increasing gradually as the test instances were scaled towards Test 8 with a memory utilisation of 1.16 GB. The increase in memory usage could be attributed to the exponential increase in the number of columns and rows generated for each given problem instance.

Test Case	Objective (R)	OPT GAP (%)	Time (sec)	Num Cols	Num Rows	Memory Consumed (GB)
Test2	276686	0	0,01	57894	47203	0,03
Test4	1019370	0	0,01	345425	276528	0,17
Test6	1820500	0	0,03	1042172	826313	0,53
Test8	2632780	0	0,06	2310938	1825748	1,16
Test10	Failed	Failed	Failed	Failed	Failed	Failed
Test12	Failed	Failed	Failed	Failed	Failed	Failed

Table 8.5: IP formulation solution validation output (loose constrained)

Test 2 started with 57894 columns and 47203 rows, while Test 8 ended with a significantly larger number of columns (2310938) and rows (1825748) to consider in the computational framework. The time required to compute the global optimal solution for each of the mentioned test cases ranged from 0.01 to 0.06 seconds which is quite remarkable. By analysing the time required to obtain the global solutions for the various test cases, it is evident that the limiting factor when employing the branch-and-bound algorithm to scaled product targeting problems is not the solution time but rather the excessive memory requirements induced by the exponential increase in the number of columns and rows. The global optimal objective function values increased from R276686 for Test 2 to R2632780 for Test 8. As noted in Table 8.5, the IP formulation could not compute any solution for test cases 10 and 12. The number of columns and rows was so overwhelming that the memory required to solve these test instances was much higher than the memory available on the computer leveraged to generate the solutions. In such a scenario, the branch-and-bound algorithm generates

and ingests the various columns and rows related to the problem instance (i.e., Test 10 and Test 12) into the solution framework. However, the memory requirements are exceeded at some point in the algorithmic process, and the code is immediately terminated to prevent a system crash. This is why no computational output is provided for Test 10 and Test 12, respectively.

For the IP formulation utilised in this thesis, the algorithm can only solve loosely constrained problem instances up to the size of Test 8. Any problem instance larger than Test 8 will terminate after a few seconds with no solution due to memory limitations. For more detail regarding the memory requirements linked to each test instance in Table 8.5, refer to Figures 8.2 and 8.3, respectively. Figure 8.2 displays the memory consumed by test cases 2, 4, and 6. The top image of Figure 8.2 represents the amount of memory utilised by Test 2. In this image, one can see that the problem size is small enough to note no significant step change in the original or swap memory. As one moves to Test 4, a small step change can be noted in the original memory consumed. This is a result of the problem size being increased incrementally. When evaluating the memory usage for Test 6, a significant step increase can be noted in the original memory while the swap memory remains untouched. Given the above data, it is apparent that there is a definite increase in the utilisation of memory as the loosely constrained optimisation problem size is scaled. However, even with scaling the problems, the IP formulation could handle test instances 2, 4, and 6 quite effectively while still having a significant amount of swap memory available.

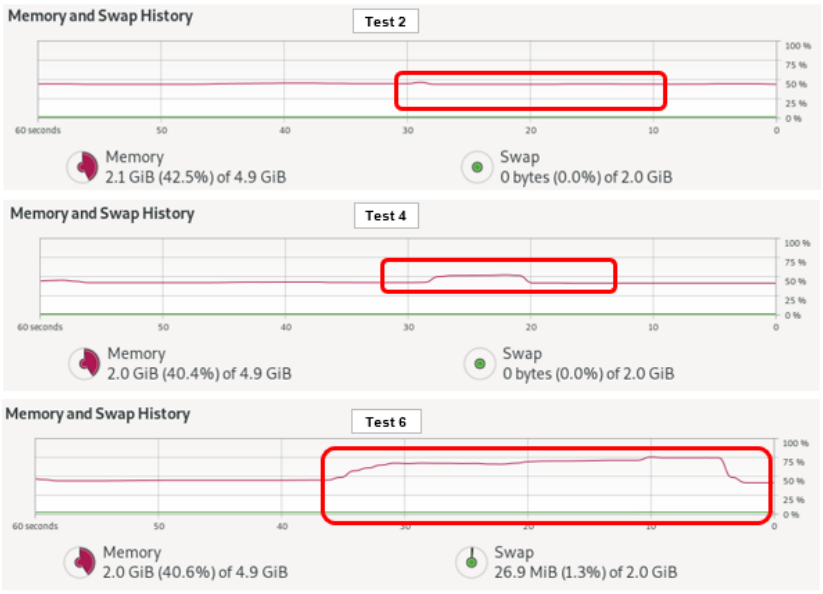

Figure 8.2: IP formulation validation tests memory consumed (loose constrained), 2–6

Figure 8.3 displays the memory consumed by test cases 8, 10, and 12. When focusing on Test 8, to solve the given optimisation problem, the entire range of original memory was consumed, with an additional 590 MB of swap memory being utilised to derive the global optimal solution. As noted in Table 8.5, Test 8 was the largest problem size that the IP formulation methodology could solve. When evaluating Test 10 and Test 12 in Figure 8.3, it is clear that each one of the problem instances required a significant amount of memory to compute the global optimal solution. In both these test cases, the original and the swap memory was consumed in its entirety, leaving no additional memory available to perform further computations. As a result, the code being executed was terminated in both instances due to the memory limitations experienced. Therefore, for Test 10 and Test 12, no solution could be derived due to early termination.

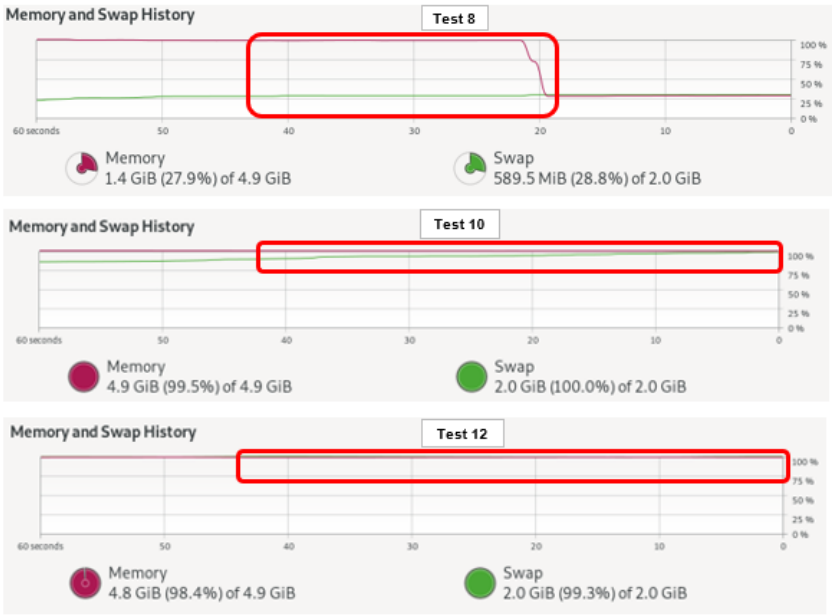

Figure 8.3: IP formulation validation tests memory consumed (loose constrained), 8–12

The preceding memory consumption analysis corresponds to the discussion relating to Table 8.5, indicating that any product targeting problem exceeding the size of Test 8 when using the IP formulation as a solution algorithm will run into memory limitations. The previous is, however, only applicable when executed on the same server size, as noted in the introduction to Chapter 7.

However, a positive aspect of the IP formulation is the short solution time required to obtain a global optimal solution, given that enough memory is available for the problem instance under consideration. In the case of having to solve a small product targeting problem within the size ranging between Test 2 and Test 8, the IP formulation will suffice. The algorithm will be able to provide the end user with a global optimal solution within a few seconds, much faster than the solution time reported for the column generation algorithm in Section 7.5.2 when applied to small compact optimisation problems. However, if large-scale problems are the aim, more than the IP formulation will be needed, and alternative solution methodologies should be considered. Section 8.2 provides information regarding the column generation validation tests and some evidence to motivate that the column generation solution algorithm will assist in solving larger-scale product targeting problems.

Test Case	Status
Test2	Success
Test4	Success
Test6	Success
Test7	Success
Test8	Success
Test9	Failed
Test10	Failed
Test11	Failed
Test12	Failed

Table 8.6: IP formulation solution validation end state

A summary of the validation test outcomes retrieved from the IP formulation is provided in Table 8.6. Similar to Table 7.14, this table provides insights into the ability of the IP formulation to solve the given optimisation test cases. At first glance, it can be noted that the branch-an-bound IP formulation successfully solved test cases 2 – 8 while failing to solve test cases 9 – 12. The detail of why the IP formulation could not solve the mentioned test instances has already been elaborated

on throughout the preceding sections.

Given that some computational limitations for the IP algorithm have been identified throughout the validation tests, a new algorithm needed to be designed throughout the thesis to outperform the current algorithm. If the IP formulation is the only available algorithm for solving the complex product targeting optimistion problem introduced in Chapter 4, it would mean that the end users would only be limited to solving small-sized problems. However, when faced with large-scale product targeting problems, the current algorithm could not generate any solution. It is for this reason that the implementation of the column generation algorithm to solve large-scale product targeting problems is tested to evaluate the effectiveness of overcoming the shortcomings of the IP formulation. Detail regarding the validation test outcomes for the column generation algorithm is provided in Section 8.2.

8.2 Novel column generation method validation results

The computational outcomes for the tightly constrained column generation validation tests are provided in Table 8.7. Similar to the results discussed in Table 7.13, this table contains information regarding the results obtained for the column generation algorithm. From the given table, it can be identified that Test 7 was the initial validation test performed using the column generation algorithm. The reported outcomes show that the column generation algorithm required 41 iterations to reach a global optimal objective function value of R 546618. When using the test results for Test 7 in Table 8.4 as a reference, it is clear that the column generation algorithm computed the same objective value for Test 7 as reported for the IP formulation. When comparing the 0.39 hours of solution time required for the column generation algorithm with the IP formulation results, it is identified that the column generation algorithm still requires a significantly longer time to compute the given solution. However, it is encouraging to see that the number of columns where reduced from 2310938 to 2000 and the rows from 1825748 to 20148 when using the column generation algorithm instead of the IP formulation. The resultant effect significantly reduces the memory requirements for Test 7. The memory consumed for Test 7 dropped from 1.16 GB to 0.1 GB, which is a tremendous computational improvement. Generating such a significant delta reduction in memory requirements already creates the possibility of solving significantly larger product targeting optimisation problems.

Evaluating the outcomes for test cases 9 and 11 in Table 8.4, it was noted that the IP formulation could not solve the two given problem instances. When applying the column generation algorithm to the mentioned test cases, solutions to these problem instances could easily be determined. Table 8.7 shows that Test 9 required 66 iterations, whereas Test 11 required 62 iterations to compute the global optimal solution. Note that there is no global optimal reference for the mentioned test instances due to the inability of the IP formulation to generate a solution for these instances. The objective function value for Test 9 equated to R560228, whereas the objective function value for Test 11 was calculated as R567408. When interpreting the integrality and optimality gap calculation results for these two problem instances, it can be concluded that the column generation algorithm reached global optimality. Test 9 reached a 2.83% optimality gap and a 1% integrality gap which is indicative of global optimality being achieved. Similar is true for Test11 with a 2.23% optimality gap and a 1% integrality gap. The time required to reach the global optimal solution for Test 9 was noted as 1.52 hours, while for Test 11, 2.37 hours were required. Similar to Test 7, the number of column and rows were impressively low, resulting in very effective memory consumption for the given test cases. Test 9 only consumed 0.05 GB, whereas Test 11 consumed 0.06 GB.

Given the enhanced capability of the column generation algorithm in solving complex product targeting problems, additional large-scale tests were conducted to try and identify the algorithm's computational limits. For this purpose, the column generation algorithm was applied in solving test instances 13, 15, and 16. As expected, the column generation algorithm solved the mentioned instances within a reasonable time frame. Test 13 required 4.98 hours to solve the optimisation

problem, whereas Test 15 consumed 12.4 hours, similar to Test 16, with a solution time of 12.08 hours. When evaluating the optimality and integrality gap calculations for each of the mentioned test cases, one can conclude that the algorithm computed a solution close to global optimality in each instance. For Test 13, an optimality gap of 0.5% and an integrality gap of 1% were achieved, indicating global optimality was obtained. The outcome for Test 15 was different because the algorithm could only compute a 10.04% optimality gap and a 1.06% integrality gap. The preceding indicates that a solution close to global optimality was computed, but the algorithm required additional run time to achieve the global optimality end state. Similar is true for Test 16 because the algorithm could compute a solution close to global optimality. Identical to what was seen for test cases 9 and 11, the memory consumed for Tests 13, 15, and 16 was minimal. In each instance, the optimisation algorithm consumed between 0.06 - 0.19 GB of memory, which is significantly lower when compared to the memory consumption profile identified for the branch-an-bound IP formulation. Again, the lower memory consumption is directly related to the minimal number of columns and rows generated by the column generation algorithm for each test case under consideration.

Test Case	Iter	MSTR OBJ (R)	Upper Limit (R)	Lower Limit (R)	INT GAP	OPT GAP (%)	Time (hrs)	Num Cols	Num Rows	Mem Cons (GB)
Test7	41	546618	569576	546618	1	4,03	0,39	2000	20148	0,1
Test9	66	560228	576559	560228	1	2,83	1,52	1675	32183	0,05
Test11	62	567408	580332	567400	1	2,23	2,37	1890	40218	0,06
Test13	35	559468	562294	559468	1	0,5	4,98	1000	60218	0,06
Test15	76	5294680	5569300	5010380	1,06	10,04	12,4	2695	80253	0,19
Test16	70	5348450	5469450	5219800	1,02	4,56	12,08	2485	100253	0,17
Test17	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed

Table 8.7: Column generation validation output (tight constrained)

The final tightly constrained test case to which the column generation algorithm was applied was Test 17. In this instance, the number of customers was scaled even further to determine the point at which the column generation algorithm will reach its computational limits for the current available server capacity. As it turns out, the column generation algorithm will only be capable of solving product targeting problems up to the size of test case 16. The algorithm will run into memory limitations when reaching tightly constrained problem sizes equal to or exceeding test case 17. The only way to solve the preceding problems will be to increase the physical memory available on the server. Without increasing the memory availability, Test 17 will pose a hard limit for the size at which the column generation algorithm will fail to generate a solution for the product targeting optimisation problem. It is, however, imperative to realise that even with the identified memory limitations, the column generation algorithm could solve tightly constrained optimisation problems that are up to five times greater than the problems that the IP formulation could solve. The preceding is a significant computational achievement and contribution to the field of operations research.

Figure 8.4 depicts the memory consumption for validation test cases 7, 9, 11, and 13. Test case 7 is portrayed at the top of the figure, and test case 13 is at the bottom. Analyzing the memory consumed over time, it is apparent that all four test cases only consumed parts of the original memory while leaving the swap memory untouched. This is significant as test case 7 for the IP formulation already started to consume quite a large quantity of swap memory. However, it is essential to note that as the problem size is iteratively increased, there is a definite step change in the amount of original memory consumed by the solution algorithm. Note that for Test 7, only 52% of the original memory is consumed, whereas, for Test 13, the original memory consumption increased to 90%. As mentioned in the previous sections, this step change is not solely related to the model memory usage but also to memory being consumed for running the virtual machine, data preparation, data ingestion, and various operational tasks. However, the preceding compounded with an algorithm that performs memory-intensive operations results in the memory limits being reached, as was seen

for the IP formulation in Chapter 8.1. Significantly reducing the algorithmic memory consumption, as was done in the case of the column generation algorithm, greatly alleviates some of those compounding effects, which in turn prevents early model termination due to memory limitations. Evidence of the mentioned improvements is reported in Figure 8.4.

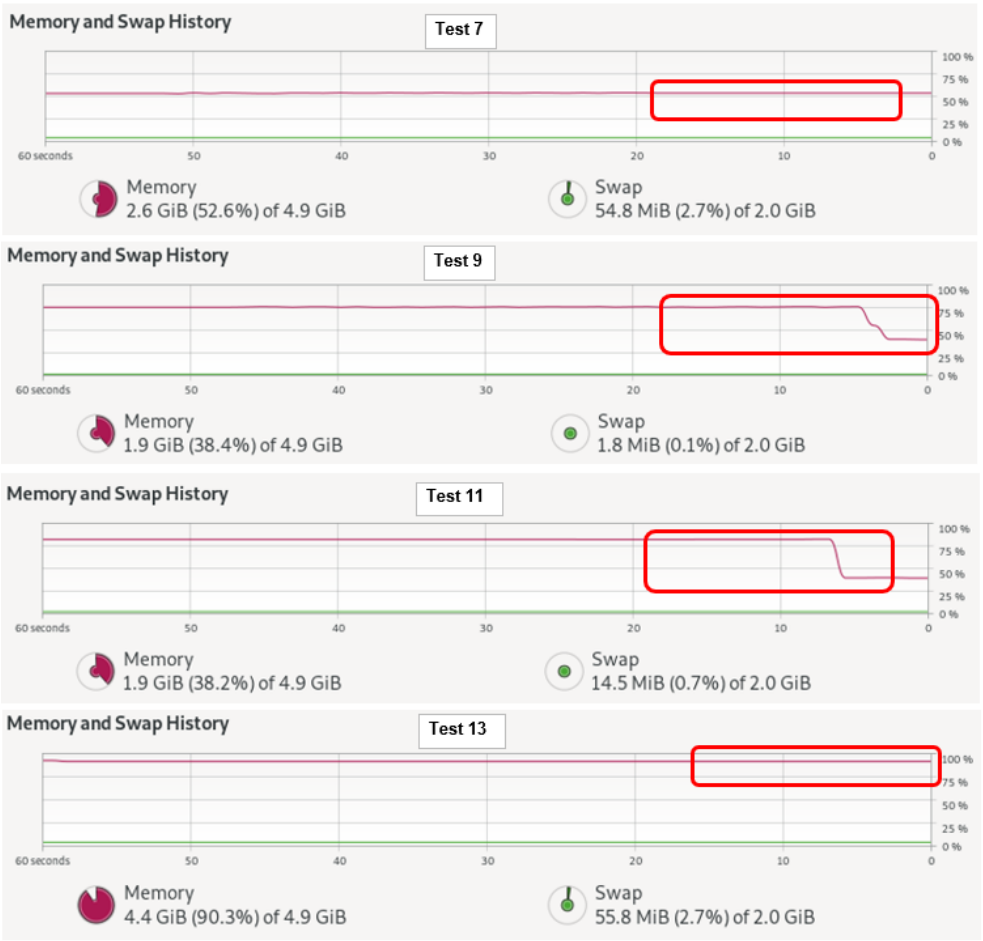

Figure 8.4: Column generation validation tests memory consumed (tight constrained), 7–13

Further memory analyses of the tightly constrained problem instances, solved using the column generation algorithm, are provided in Figure 8.5. The test cases in this figure include test cases 15, 16, and 17, with test case 15 presented by the top image and test 17 by the bottom image, respectively. For these three test instances, one can already note that the memory requirements are becoming increasingly stringent and that the optimisation model will not be able to solve significantly larger problems than those it was presented with.

In Test 15, the operational process is already consuming 92.3% of the original memory while also leveraging an additional 65.1% of the available swap memory. When moving to Test 16, it is apparent that the server is already running at peak operational capacity of 96% original memory as well as 97.5% swap memory. Therefore, when the model is applied to Test 17, no additional memory is available to account for the additional model requirements, and the computational process is terminated. This, in turn, results in no solution being computed for Test 17. When considering the preceding information, the operational limits for the column generation algorithm, when applied too tightly constrained product targeting problems, were identified. In order to evaluate the column generation algorithm's capability in solving loosely constrained product targeting problems, tests are performed for which the results are reported in Table 8.8.

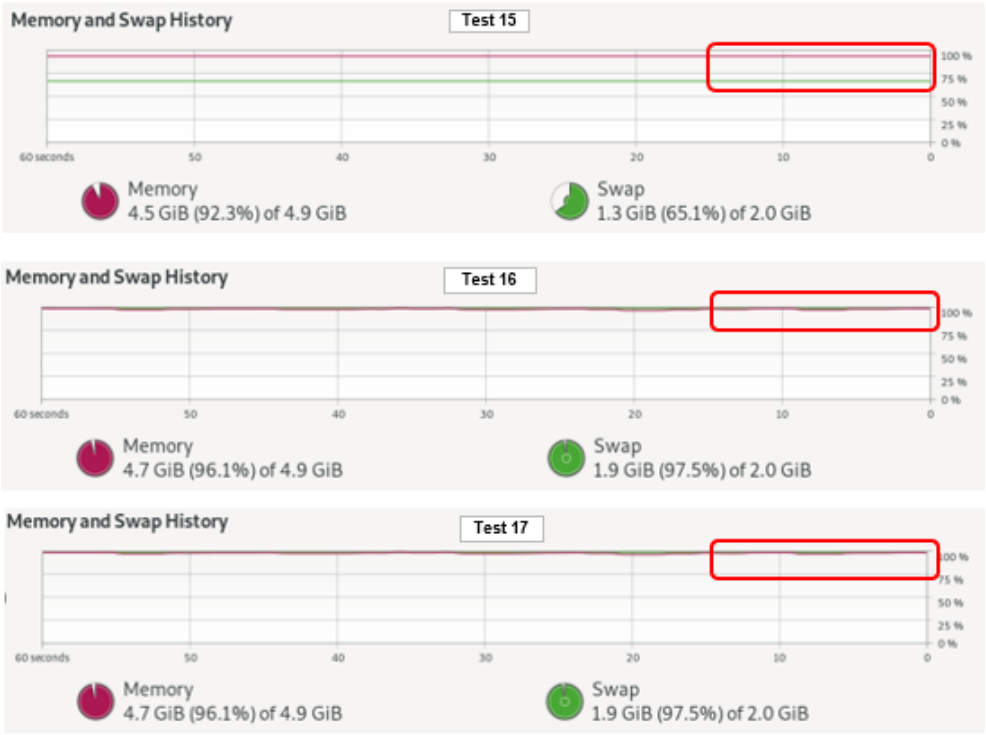

Figure 8.5: Column generation validation tests memory consumed (tight constrained), 15–17

As stipulated during the introduction to Chapter 8, the problem instances reported in Table 8.3 were each split into two separate testing streams (A and B) to test the ability of various column generation algorithmic formulations in solving loosely constrained problem instances. The results of the various testing streams related to testing cases 2, 4, 6, 8, 10, and 12 can be seen in Table 8.8. As previously mentioned, testing steam A is the application of the standard column generation formulation to the various test instances. In contrast, test stream B represents the results obtained after implementing the algorithmic modifications stipulated in Section 6.3.8. As an initial analysis, the focus is first set on test stream A. When evaluating the ability of the column generation algorithm to solve test cases 2 – 12 as stipulated in Table 8.8, it is clear that in each instance, the column generation algorithm was able to compute an answer close to global optimality. Given that the loosely constrained optimisation problems are more complex to solve due to the increased search space, the column generation algorithm required significant computing hours to try and reach global optimal solutions. For this reason, the compute time was limited to 12 hours in some instances to prevent excessive run time while still portraying the ability of the column generation algorithm to generate close to global optimal solutions within a reasonable time frame.

For Test 2_A, the compute time was reported as 6.93 hours. Test 4_A and Test 6_A were allowed to run their computational processes for 48 hours before terminating, while the remainder of the test cases were terminated at the 12-hour mark. When comparing the results obtained for testing stream A in Table 8.8 with that of Table 8.5, it is apparent that the column generation algorithm could only reach global optimality for Test 2_A (having the same objective function values as the IP formulation solution). For the remainder of the loosely constrained test cases within stream A, only near global optimal solutions could be computed within the allowable time frame. The excessive computational time could be attributed to the extensive computational overhead of the column generation algorithm that had to be managed. Even though these test instances took quite a while to compute a solution, the number of rows and columns and the memory consumed by the optimisation algorithm for each test instance in stream A was significantly less than that reported for the IP formulation in Table 8.5. The number of columns generated for the test instances in stream A ranged from 2580 to 3750, whereas the number of rows was calculated to be between 2043 to 40218. The memory consumed by the optimisation algorithm for each test case did not exceed

a total of 0.28 GB, with each test case performing below the stipulated mark. The optimality gap for Test 2_A, Test 8_A, Test 10_A, and Test 12_A was computed around the 10% – 15% mark. Test 4_A and Test 6_A could not attain optimality gaps within the 10% – 15% region and ended up with optimality gaps equating to values of 51.9% and 34.93%, respectively. When evaluating the number of iterations required to compute a near global optimal solution for each test case, it seems like the larger the problem instance is scaled, the fewer iterations are required to compute a global optimal solution when applying the standard column generation algorithm.

Test Case	Iter	MSTR OBJ (R)	Upper Limit (R)	Lower Limit (R)	INT GAP	OPT GAP (%)	Time (hrs)	Num Cols	Num Rows	Mem Cons (GB)
Test2_A	515	276686	303948	276349	1	9,08	6,93	2580	2043	0,09
Test2_B	589	276521	309398	269492	1,03	12,9	0,52	447	2043	0,03
Test4_A	338	940634	1770000	850246	1,11	51,9	48,3	3390	6078	0,23
Test4_B	1989	1020000	1140000	871805	1,17	23,62	47,87	229	6078	0,1
Test6_A	249	1730000	2380000	1550000	1,12	34,93	48,43	3750	12113	0,28
Test6_B	1169	1815380	2008500	1573180	1,15	21,67	47,71	1036	12113	0,42
Test8_A	167	2621900	2829550	2367240	1,11	16,34	12,1	3360	20148	0,18
Test8_B	267	2626010	4379560	2381470	1,1	45,62	12,05	1643	20148	0,23
Test10_A	129	3574660	3821870	3332550	1,07	12,8	12,14	3250	32183	0,17
Test10_B	167	3574050	4636860	3334410	1,07	28,09	12,11	2001	32183	0,24
Test12_A	110	4648410	4938370	4368700	1,06	11,54	12,15	3330	40218	0,18
Test12_B	139	4652750	4961520	4369700	1,06	11,93	12,1	2466	40218	0,24
Test14	33	2185730	3664650	2185730	1	94,04	2,77	Failed	Failed	Failed

Table 8.8: Column generation validation output (loose constrained)

When evaluating the results reported for test stream B where the algorithmic changes as stipulated in Section 6.3.8 were applied, it could be noted that there is a definite change in the dynamics of the algorithm. Firstly, the number of columns was significantly reduced compared to the number of columns generated for the same test cases as in stream A. This is because the algorithm discussed in Section 6.3.8 is designed to eliminate unused columns as the algorithm progresses through the computational process. A reduction in the number of columns at each iteration results in the problem size (MB) being reduced and the time spent on a single iteration being decreased. This, in turn, either results in the modified column generation algorithm being able to perform more iterations within a given period when compared to the standard column generation algorithm or within a shorter period reaching the same amount of iterations as that of the standard column generation algorithm. When comparing Test 4_A and Test 6_A with Test 4_B and Test 6_B, the modified column generation algorithm outperforms the standard column generation algorithm concerning its ability to reach global optimality. Test 4_A obtained an optimality gap of 51.9%, whereas Test 4_B reached an optimality gap of 23.62%. Similar is true for Test 6_A, which could only attain a gap of 34.93%, whereas Test 6_B achieved an optimality gap of 21.67%. However, when evaluating Test 2, Test 8, Test 10, and Test 12, the modified algorithm is outperformed by the standard column generation algorithm. The standard column generation algorithm was able to attain a better gap percentage and higher objective function value in the majority of the cases and utilise fewer resources due to the reduced number of iterations. One obvious thing is that removing columns from the original column generation algorithm resulted in a decay of the "good" columns being introduced at each iteration. This, in turn, forced the algorithm to run through various additional iterations to try to make up for the loss in solution accuracy. In conclusion, in some cases where the standard column generation algorithm cannot get close to the global optimal solution, the modified algorithm can improve significantly on the solution being computed. However, if the standard column generation algorithm can reach a solution close to global optimum, the modified version will not outperform the proposed solution. The implementation of the various column generation algorithms and their modifications should therefore be evaluated on the face value of the problem to be solved, and a decision should be made accordingly.

The last loosely defined test case, solved using the column generation algorithm, was Test 14. In this instance, the problem size was enlarged by increasing the number of customers to 15000. The preceding was done to determine the size limit where the column generation algorithm will fail to find a solution. Table 8.8 contains the results for the mentioned test case. Note that the algorithm was able to initiate the computational process; however, at around 33 iterations and an optimality gap of 94%, the problem terminated on memory limitations. For Test 14, an initial feasible solution could be obtained. However, the algorithm could not come anywhere near the global optimal solution. As a result, Test 14 was identified as the hard limit for the problem size where the column generation algorithm would reach its memory limitations. When trying to solve the same or larger-sized loosely constrained product targeting problems, the column generation algorithm will be incapable of generating a sensible solution. However, similar to the analysis performed for the tightly constrained problems, it should also be mentioned that even though the column generation algorithm reached some memory limitations, it was still able to solve problem sizes double the size of what the branch-and-bound algorithm was capable of solving. The preceding is quite a significant contribution to the operations research community.

Figures 8.6 and 8.7 provide insights into the memory consumption analyses performed for the various loosely constrained validation test cases solved using the column generation algorithmic framework. When analysing the information noted in figures 8.6 and 8.7, it is apparent that when using the column generation algorithm to solve product targeting optimisation problems, the memory consumed from an overall operational perspective is minimal. For Test 2_A and Test 2_B, only 42.9% and 43.9% of original memory were consumed without touching the swap memory. Similar is true for Test 4_A, and Test 4_B, with these test instances consuming around 45.4% of the original memory, slightly higher than the memory consumption noted for test instance 2. The overall memory consumed for steam A versus stream B for the various test instances does not vary significantly. At a point in time, it is seen that the standard column generation algorithm and the modified column generation algorithm discussed in Section 6.3.8 consume the same amount of memory from an overall operational perspective. When evaluating the memory consumed for test instances 6, 8, 10, and 12, it becomes apparent that memory usage steadily increases as the problem instances are scaled.

Figure 8.6: Column generation validation tests memory consumed (loose constrained), 2A–8B

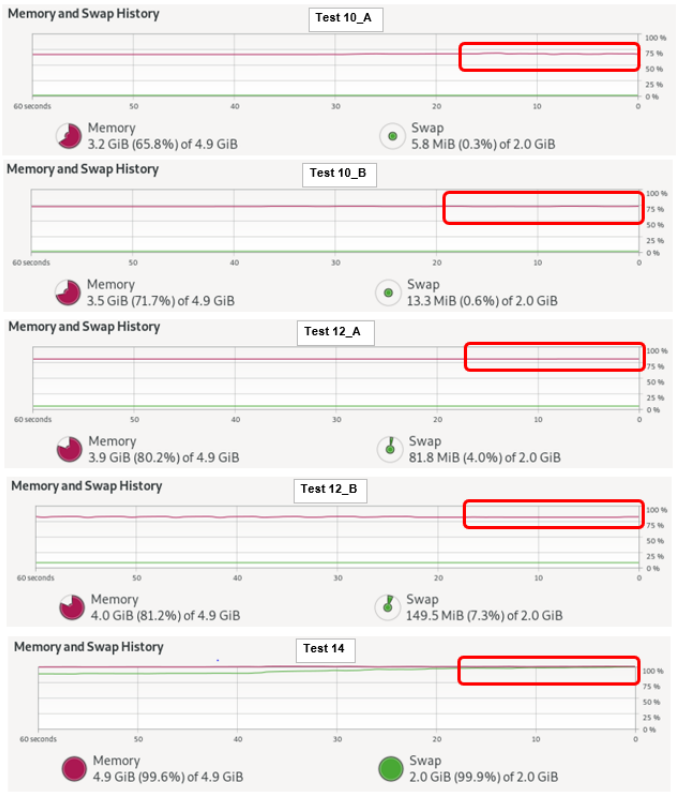

Figure 8.7: Column generation validation tests memory consumed (loose constrained), 10A–14

Test instance 6 seems to be using around 51% of the original memory, with that usage increasing steadily up to a point where Test 12 consumes around 81.2% of the original memory. It is interesting to note that the memory consumption curves for the column generation algorithms are very smooth. In contrast, the IP formulation has a rapid, immediate step change in memory consumption, whereafter, the algorithm finds a solution or terminates on memory limitations. Test 14 consumes all available original and swap memory from an overall operational perspective, after which it terminates on memory limitations.

Table 8.9 summarizes the outcomes of the various test cases solved as part of the validation tests for the column generation algorithm. At first glance at the summary table, it is clear that the column generation algorithm was able to compute solutions for the majority of the test instances. However, the column generation algorithm failed to find solutions for test instances 14 and 17. It is, however, essential to note that for those test instances where solutions were computed, for some, the global optimal solution was obtained. In contrast, for others, only a solution close to global optimality was computed. From the data provided in this table, it is also apparent that the column generation algorithm solved significantly more product targeting optimisation problems compared to the IP formulation, as seen in Table 8.6. The IP formulation solved five product-targeting problems of varying sizes. In comparison, the column generation algorithm solved 12 product targeting problems with much greater search spaces, 7 more than the IP formulation.

Considering the information provided throughout Chapters 7 and 8 regarding the verification and validation test outcomes, the column generation algorithm proposed throughout this thesis outperforms the IP formulation for large-scale product targeting problems. Furthermore, the column generation algorithm can find solutions for problem sizes where the IP formulation cannot even fit the initial problem into memory. It was, however, identified that the contrary is true when solving small-scale product targeting problems. In this case, the IP formulation would be the preferred solution as it provides the end user with the global optimal solution within a few seconds.

The column generation algorithm also provides the end user with the global optimal solution. How-

ever, this algorithm takes significantly longer to arrive at the desired end state. As a result, the end user should evaluate the size and complexity of the given product targeting problem to be solved and use the preceding as guidelines to decide which one of the proposed algorithms to commit.

Test Case	Status
Test2_A	Success
Test2_B	Success
Test4_A	Success
Test4_B	Success
Test6_A	Success
Test6_B	Success
Test7	Success
Test8_A	Success
Test8_B	Success
Test9	Success
Test10_A	Success
Test10_B	Success
Test11	Success
Test12_A	Success
Test12_B	Success
Test13	Success
Test14	Failed
Test15	Success
Test16	Success
Test17	Failed

Table 8.9: Column generation validation end state

To provide the reader with a summarised view of how the various models performed while applying them to the verification and validation test cases, insights derived from the raw data tables, as provided throughout Chapters 7 ad 8, are compiled in Chapter 9. Specific emphasis is placed on the memory utilisation of each algorithm as well as the ability of the algorithms to compute global optimality gaps.

Chapter 9

Model performance comparison

In Chapter 9, comparative insights are provided regarding the performance of the IP formulation versus the column generation algorithms when applied to various sizes of product targeting optimisation problems. After proving the dominance of the column generation method, various aspects of this algorithm, such as testing the model performance when incorporating variations in starting algorithms, as well as the addition and exclusion of parallel processing technology and dual smoothing techniques, are explored.

9.1 Memory utilisation

Figure 9.1 provides a summarised view of the memory consumed by both the IP formulation and the column generation algorithm for test instances 1, 3, 5, 7, 9, 11, 13, 15, and 16.

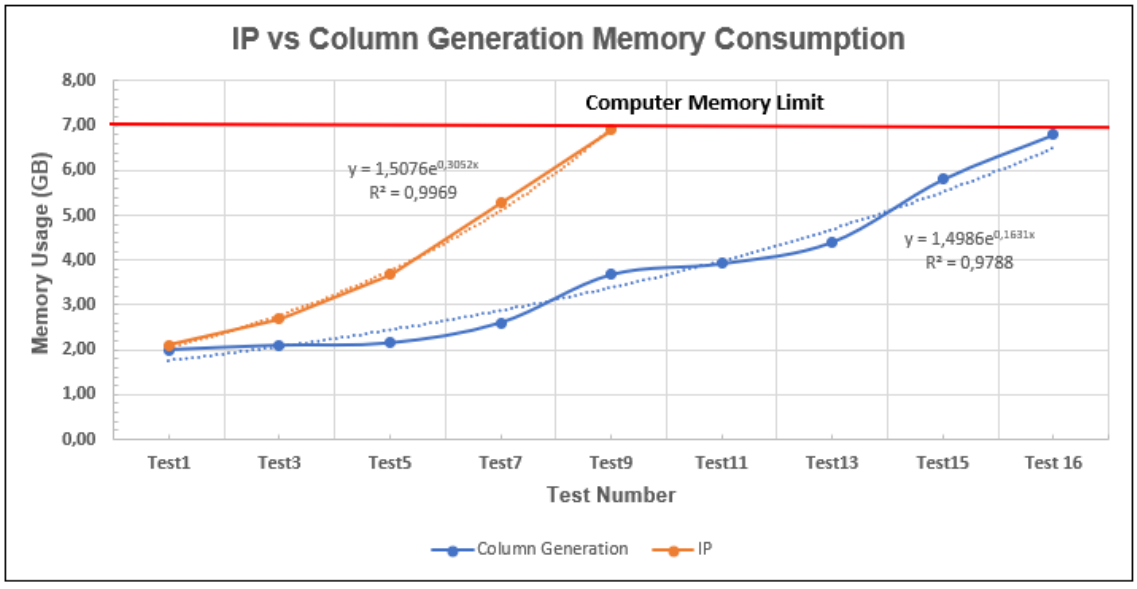

Figure 9.1: IP vs column generation memory consumption

When following the memory utilisation profile presented for the IP formulation (represented by the orange curve), it is apparent that the algorithm operated close to the server's memory limits during the computation of the solution for test case 9. As a result, the IP formulation could not compute the global optimal solution for Test 9, as seen in Table 8.6, due to memory limitations being reached. However, the IP formulation could operate well below the memory limits for test cases 1, 3, 5, and 7 without running into any memory limitations and, as a result, could compute the desired global optimal solutions. From the mentioned analysis, the IP formulation follows an exponential memory consumption trend when scaling the optimisation problem instances to be solved.

The blue line represents the memory utilisation profile for the column generation algorithm in Figure 9.1. When evaluating the blue trend line, it is clear that the column generation algorithm only started operating at the server's memory limits during the execution of test instance 16. Note that the column generation algorithm could still solve Test 16, as seen in Table 8.9. It is reasonable to assume that the column generation algorithm will not be able to solve problem instances exceeding the size of Test 16 as this test instance already consumed almost all of the available memory. When comparing the reported memory profile of the column generation algorithm with that of the IP formulation, it is clear that there is a significant difference between the two curves. By implementing the column generation algorithm, the end user achieved a large delta drop in memory consumption allowing the algorithm to solve significantly larger product targeting optimisation problems. For example, for Test 5, the IP formulation consumed approximately 4 GB of memory, whereas the column generation algorithm only consumed 2 GB. Similar is true for Test 7, with 5.2 GB of memory being utilised by the IP formulation, where the memory requirements for the column generation algorithm were around the 2.7 GB mark. The preceding are just two examples noted in Figure 9.1, with the remainder following the same trend.

9.2 Optimality gap

When evaluating the performance of various algorithms, it does not warrant only looking at memory utilisation. For example, one could have an excellent algorithm for memory utilisation, but the algorithm is not capable of achieving global optimality. It is for this reason that an analysis was done on the ability of the IP formulation as well as the column generation algorithm in achieving global optimality (optimalitiy gap equating to 0%) when attempting to solve each of the test cases defined in Chapters 7 and 8. The top image depicted in Figure 9.2 represents the optimality gap computed for each of the tightly constrained test cases while leveraging both the IP formulation and the column generation methods as solution algorithms. Similar is valid for the bottom part of the figure. However, this image relates to the loosely constrained test cases and their respective outcomes.

When analysing the results provided in Figure 9.2 related to the tightly constrained test instances 1, 3, 5, and 7, one notes that both the IP formulation and the column generation algorithm could compute answers within the 0% – 10% optimality gap range. When moving towards test instance 9, it is apparent that the IP formulation could not compute any solution for the given problem instance, similar to what was mentioned in previous sections. Therefore the instance was assigned an optimality gap equal to 100%. The same is true when evaluating the computational outcomes for test instances 11, 13, 15, and 16. In each of the preceding instances, the IP formulation could not compute an answer, and as a result, no gap could be calculated (a value of 100% was assigned). When considering the results generated for test cases 9, 11, 13, 15, and 16 when using the column generation method as a solution algorithm, it is identified that for each of the mentioned cases, an optimality gap between 0% – 10% could be computed. When comparing the ability of the two algorithms to compute an optimality gap for tightly constrained product targeting problems, it is evident that the column generation algorithm outperforms the branch-and-bound algorithm quite significantly.

For the loosely constrained test cases depicted in Figure 9.2, a similar trend to the tightly constrained test cases is noted when looking at the optimality gap results. The IP formulation can compute a 0% optimality gap for test cases 2, 4, 6, and 8. Only after test case 8 (problem instances 10, 12, and 14) does the IP formulation start to run into memory limitations resulting in no gap being computed. When evaluating the results generated for the column generation algorithm, it is noted that in each of the loosely constrained test cases, the column generation algorithm was able to compute a solution and, consequently, an optimality gap.

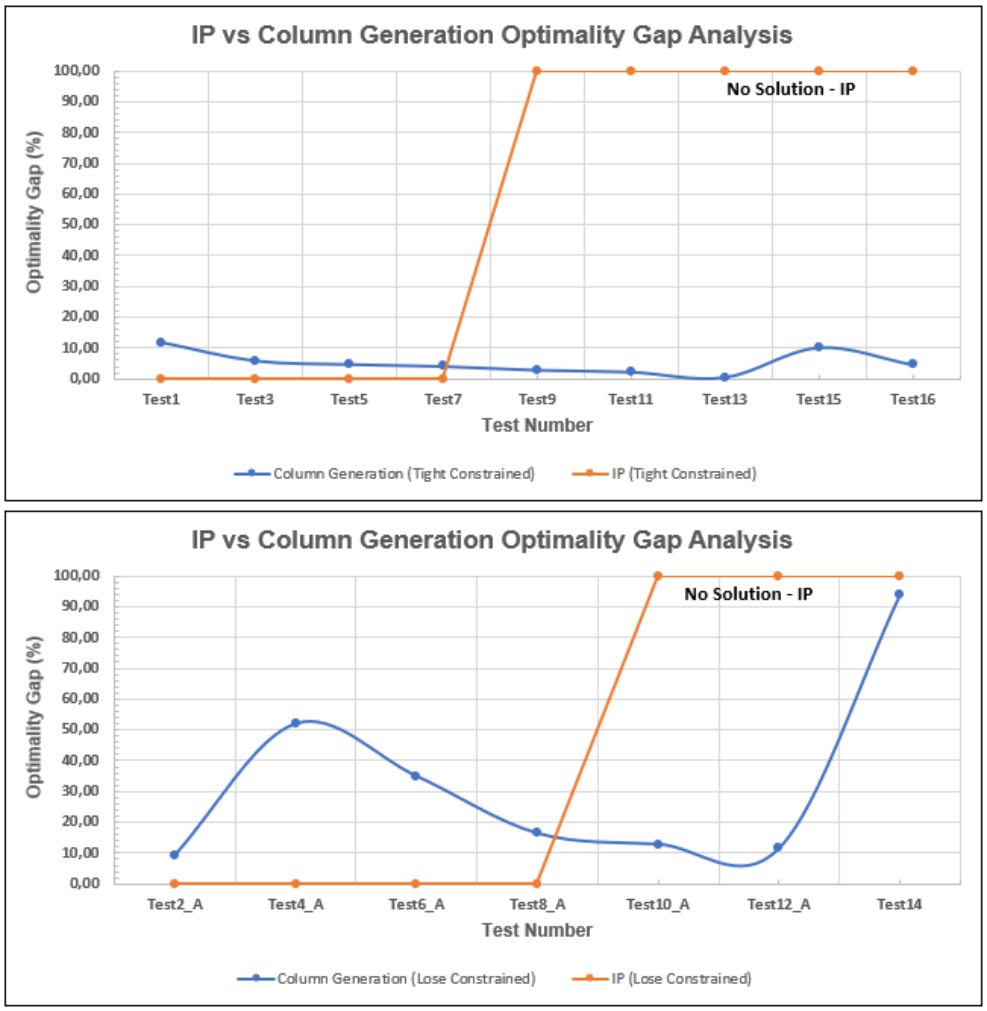

Figure 9.2: IP vs column generation optimality gap

However, the column generation algorithm seems to be less efficient in solving the loosely constrained problem instances as optimality gaps ranging between 10% – 50% were reported for the various problem instances. Even though this is the case, the column generation algorithm could still compute acceptable solutions for the problem instances where the IP formulation failed to do so. Performing a comparative analysis of the outcomes of both these algorithms, the branch-and-bound algorithm would once again be the better algorithm to utilise when trying to solve small-sized, loosely constrained product targeting optimisation problems. The column generation algorithm will outperform the IP formulation if the end-user wants to solve medium to large product targeting problems.

9.3 Row and columns added to models

When applying the column generation theory to solve sizeable complex optimisation problems, the computational gain achieved by the column generation algorithm can be attributed to the reduction in the number of columns and rows considered to derive the global optimal solution. The column generation algorithm only considers a subset of the extreme points and rays for a given problem, thus reducing the memory and computational requirements to solve the given problem. An analysis is performed on the number of columns and rows generated from the verification and validation tests to confirm the potential benefit. Given that the number of rows and columns generated for each test case stipulated throughout Chapters 7 and 8 vary in order of magnitude, a logarithmic scale is utilised to plot the test outcomes on the same graphing axis to improve readability. The data depicted in Figure 9.3 represents the number of rows and columns generated for the tightly constrained test cases when applied to both the IP formulation and the column generation algorithm. The top

image in Figure 9.3 depicts the number of columns generated by each of the mentioned algorithms. In contrast, the bottom image is focused on the number of rows that originated from the computational process. When assessing the top image, it is apparent that the number of columns generated by the IP formulation was significantly more when compared to the column generation algorithm. For Test 1, we note that the number of columns generated by the IP formulation equated to a value of 57894. For the same test case, the column generation algorithm outperformed the IP formulation to a large extent by generating only a total of 755 columns. A similar trend is noted for Test 3, with the IP formulation generating 345425 columns while the column generation algorithm could compute a solution using only a minimum of 950 columns. For test cases 5 and 7, it is once again noted that the column generation algorithm can generate significantly fewer columns when compared to the IP formulation. For test cases 9, 11, 13, 15, and 16, it is apparent that the branch-and-bound formulation could not compute any solution, as mentioned throughout the preceding sections, and therefore no columns could be generated. The column generation algorithm was, however, capable of computing a solution for each of the mentioned test cases while generating a limited amount of columns ranging between 1000 – 2700 for the respective test cases. Given the preceding analysis, it is apparent that the column generation algorithm can drastically reduce the number of columns compared to the columns being generated when using the IP formulation as a solution methodology. This, in turn, allows the column generation algorithm to reduce computational requirements significantly, which enables the algorithm to handle remarkably larger optimisation problems.

The bottom image of Figure 9.3 provides results regarding the number of rows generated for each of the tightly constrained test cases discussed in Chapters 7 and 8. Similar to the analysis performed on the number of columns being computed, it is seen that the IP formulation produced a large extent of additional rows for each of the given test cases when compared to the rows that were generated by the column generation algorithm for the same test cases. When considering the data for Test 1, it is apparent that the IP formulation for this test instance generated 47203 rows. In contrast, the column generation algorithm could keep the number of rows to a mere 2043 for the same instance. For test cases 3, 5, and 7, the IP formulation generated 276528, 826313, and 1825748 rows, respectively. When evaluating the results obtained after applying the column generation algorithm, it is noted that it generated only 6078, 12113, and 20148 rows for the same test instances. The preceding indicates that the column generation algorithm also significantly outperforms the IP formulation when it comes to the number of rows generated for each test instance. The ability of the column generation algorithm to maintain the number of rows being generated to a minimum enables the algorithm to solve significantly larger optimisation problems than the capability of the IP formulation. The aforementioned is clear when focusing on test instances 9, 11, 13, 15, and 16, with the IP formulation not being able to calculate a solution as a result of a large number of columns and rows present, resulting in an excessive amount of compute capacity/ memory required to solve the various problems. In the process of evaluating the results obtained for test cases 9, 11, 13, 15, and 16 after applying the column generation algorithm, it is apparent that the number of rows generated for each instance was kept to a minimum with Test 9 consisting of 32183 rows, Test 11 having 40218, Test 13 containing 60218, Test 15 ending with several 80253 rows and lastly Test 16 not exceeding 100253 rows. The preceding analysis indicates that for tightly constrained product targeting problems, the column generation algorithm can significantly reduce the number of columns and rows required to compute the global optimal solution. On the other hand, the IP formulation cannot consider only a subset of the tightly constrained columns and rows and is designed to consider all available columns and rows for a given problem within memory. Therefore, the former can solve significantly larger optimisation problems, whereas the latter runs into memory limitations for the same problem instances.

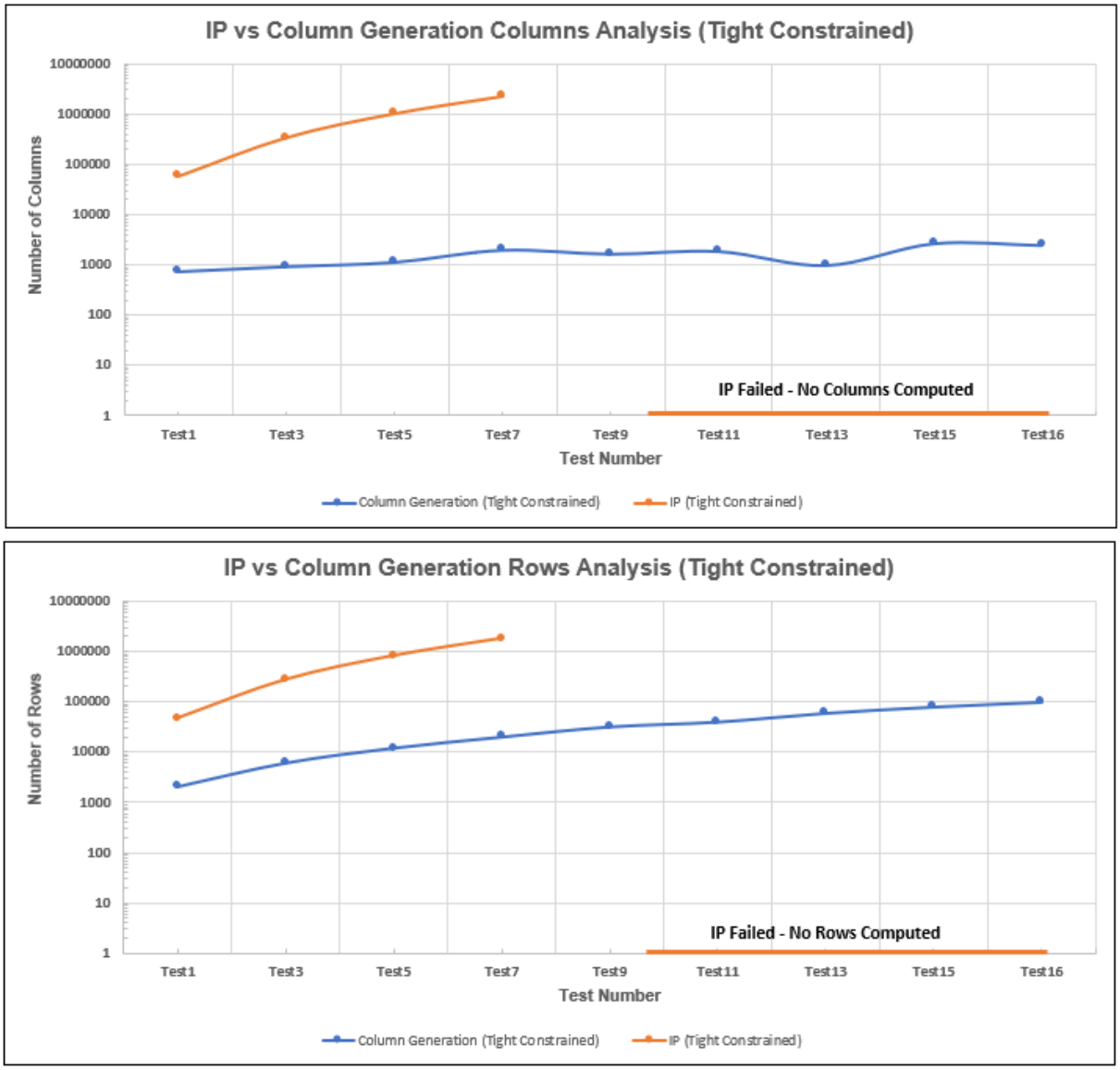

Figure 9.3: IP vs column generation column and row nalysis (tightly constrained)

A similar column and row generation analysis is performed for the loosely constrained test cases reported in Chapters 7 and 8, with the results being depicted in Figure 9.4. The top image of Figure 9.4 represents the number of columns generated by both the IP formulation and column generation algorithms. In contrast, the bottom image portrays the number of rows computed by the above-mentioned algorithms.

In each of the images provided in Figure 9.4, it is clear that for the loosely constrained optimisation problems, the column generation algorithm can also outperform the IP formulation regarding the number of columns and rows being generated. We can see that for test instances 2, 4, 6, and 8; the IP formulation generated 57894, 345425, 1042172, and 2310938 columns and a total of 47203, 276528, 826313, and 1825748 rows for each of the respective test instances. When comparing the preceding with the results generated for the column generation algorithm, it is noted that the number of columns and rows produced is significantly reduced. For test instances 2, 4, 6, and 8, the column generation algorithm only generated 2580, 3390, 3750, and 3360 columns, whereas the number of rows equated to 2043, 6078, 12113, and 20148 for each of the respective problem instances. As mentioned throughout the preceding sections, the IP formulation could not compute a solution for test instances 10 and 12 due to the excessive number of columns and rows considered within this solution methodology. Due to the early termination of the algorithm, the exact number of columns and rows could not be computed for the mentioned test instances.

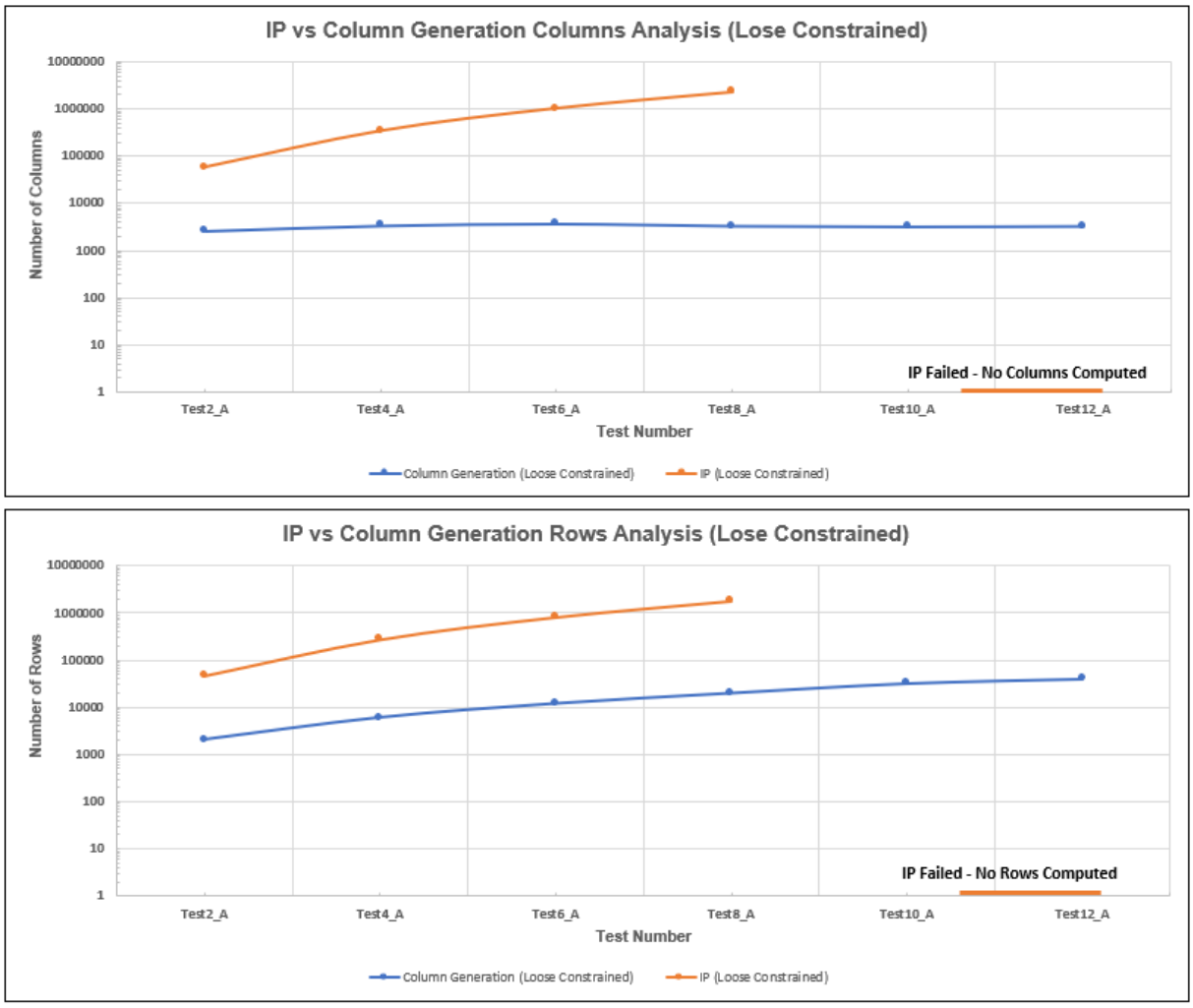

Figure 9.4: IP vs column generation column and row analysis (loosely constrained)

However, the column generation algorithm was able to compute a solution consisting of only 3250 columns and 32183 rows for test instance 10 and 3330 columns and 40218 rows for problem instance 12. The preceding columns and rows generated are still significantly lower than those produced for test 7 when considering the IP formulation. As a result, the column generation algorithm can solve significantly larger loosely constrained optimisation problems than the IP formulation due to its capability to reduce the number of columns and rows required to compute the global optimal solution. Therefore, the column generation algorithm seems superior to the IP formulation when it comes to limiting the number of columns and rows being generated for both tightly and loosely constrained optimisation problems.

9.4 Column generation starting algorithm testing

During the discussion of the column generation framework, as noted in Section 6.3.1, it was mentioned that it leverages a staring heuristic algorithm to generate an initial feasible solution. Multiple avenues were explored as part of developing and designing the given starting heuristic. One was an altered version of the IP formulation, and the other was based on a greedy heuristic algorithmic approach. For detail regarding the two algorithms, refer to Chapter 6, Section 6.3.1. In order to evaluate the capability of each of the mentioned starting heuristic methodologies in assisting the column generation algorithm in computing a global optimal solution, some test scenarios were compiled, which are reported in Table 9.1. First, an initial test case was formulated leveraging 500 customers, 5 products, and 3 channels to test the performance of each starting heuristic. After that, the test instances were iteratively scaled to evaluate the starting heuristics' ability to generate initial

feasible solutions for large product targeting problems. Test cases 7, 10, and 12 consisted of 5000, 8000, and 10000 customers, respectively, while considering 20, 25, and 30 products for the various instances. The number of communication channels incorporated into the preceding test cases was kept constant at 3.

Test Case	Model Type	Num Cust	Num Prod	Num Chan	Type Constrained
Test1	DM	500	5	3	TIGHT
Test1_IP	DM	500	5	3	TIGHT
Test7	DM	5000	20	3	TIGHT
Test7_IP	DM	5000	20	3	TIGHT
Test10_A	DM	8000	25	3	LOOSE (Kept Cols)
Test10_D_IP	DM	8000	25	3	LOOSE (Kept Cols)
Test12_A	DM	10000	30	3	LOOSE (Kept Cols)
Test12_D_IP	DM	10000	30	3	LOOSE (Kept Cols)

Table 9.1: Column generation starting algorithm input

Test instances 1 and 7 were formulated as tightly constrained optimisation problems, whereas test cases 10 and 12 were designed as loosely constrained problems. The above test cases were formulated in an attempt to capture the full capability of the starting heuristic algorithms by applying them to varying degrees of complex optimisation problems while recording the computational outcomes. The preceding was then utilised to select the best possible starting heuristic from the available algorithms to form part of the overall column generation framework.

The results obtained from the various test instances in Table 9.1 are provided in Table 9.2. For Test 1, the column generation algorithm was able to compute a global optimal solution leveraging the greedy algorithm as starting heuristic. The master objective function value, which was computed, equated to a value of R212448, while the upper and lower bounds took on values of R240935 and R212448, respectively. It took the column generation algorithm 0.08 hours to compute the solution while only generating 1270 columns and 2043 rows. The mathematical model for this test instance consumed a total of 0.01 GB of memory. Results for Test 1_IP were generated after utilising the modified IP formulation as starting heuristic. In this instance, the column generation framework performed equally compared to Test 1. One thing that was noted is that the starting heuristic did not significantly influence the column generation algorithm performance but only enabled the column generation algorithm to be initialised. If an initial feasible solution can be obtained from any starting heuristic, the column generation algorithm can take over and compute the global optimal solution. The same principle is noted when evaluating the results obtained for Test 7 (leveraging the greedy algorithm as starting heuristic) and Test 7_IP (employing the modified IP formulation as starting heuristic). In both instances, the column generation algorithm computed a master objective function value of R546618 while computing a lower bound of R546618 for both test instances. The upper bound for Test 7 equated to a value of R569572, while the upper bound for Test 7_IP was computed at R568021. The remainder of the results for these test instances were practically identical, with both algorithms generating global optimal solutions within approximately 4 hours, having 2000 columns, 20148 rows, and consuming about 0.10 GB of memory.

When evaluating the results obtained for the large loosely constrained optimisation problems in Table 9.2, it is apparent that no solutions could be computed for Test 10_D_IP and Test 12_D_IP. The limiting factor in these scenarios was the usage of the modified IP formulation as starting heuristic to generate initial feasible solutions. Given that the mentioned heuristic is still required to ingest all available columns and rows of the optimisation problem into memory (based on the branch-and-bound framework) before generating an initial feasible solution, it becomes an issue for larger problems as the memory required to compute the solution will eventually exceed the available server capacity. The inability to obtain answers for Test 10_D_IP and Test 12_D_IP are good examples of the preceding. Even though no solutions could be obtained for the mentioned test instances, it was identified that when leveraging the greedy algorithm as starting heuristic to

the column generation algorithm for the same test instances, close to optimal solutions could be obtained.

Test Case	Iter	MSTR OBJ (R)	Upper Limit (R)	Lower Limit (R)	INT GAP	OPT GAP (%)	Time (hrs)	Num Cols	Num Rows	Mem Cons (GB)
Test1	150	212448	240935	212448	1	11,82	0,08	1270	2043	0,01
Test1_IP	153	212448	240324	212448	1	11,6	0,1	1270	2043	0,01
Test7	139	546618	569572	546618	1	4,033	4,40	2000	20148	0,10
Test7_IP	137	546618	568021	546618	1	3,77	4,03	2000	20148	0,10
Test10_A	129	3574660	3821870	3332550	1,07	12,80	12,14	3250	32183	0,17
Test10_D_IP	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed
Test12_A	110	4648410	4938370	4368700	1,06	11,54	12,15	3330	40218	0,18
Test12_D_IP	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed

Table 9.2: Column generation starting algorithm output

Note that for Test 10_A, an objective function value of R3574660 was obtained within 12,14 hours while leveraging only 3250 columns and 32183 rows. Similar is true for Test 12_A, where an objective function value of R4648410 was calculated within 12.15 hours. A total of 3330 columns and 40218 rows were generated for the given problem, with the mathematical model only consuming 0.18 GB of memory. The above test results show that the greedy algorithm, as mentioned in Section 6.3.1, would be the preferred starting heuristic algorithm when combined with the column generation algorithm. The preceding algorithm would allow the end user to solve significantly larger optimisation problems. The modified IP formulation starting heuristic would just not be capable of handling large-scale product targeting problems. It would, as a result, render the column generation algorithm useless. This is why the greedy algorithm was chosen within this thesis to form part of the proposed mathematical framework discussed in Section 6.3.

Figure 9.5 provides information regarding the overall memory usage for the column generation algorithm using the modified IP formulation as starting heuristic for each of the above-mentioned test cases. Note that the column generation algorithm using the greedy algorithm as starting heuristic is not provided here, as the associated memory response curves were already discussed throughout the former sections. In the given figure, it is clear that the modified IP formulation was able to generate an initial feasible solution for Test 1_IP and Test 7_IP, allowing the column generation algorithm to compute a global optimal solution for each of the test instances. Note that for Test 1_IP, a total of 41.4% original memory was consumed with no swap memory utilised. Similar is true for Test 7_IP, with 71.3% of the original memory being utilised while only 2.9% of the swap memory was committed. The increase in memory requirements between Test 1_IP and Test 7_IP resulted from the optimisation problem size. It is, however, apparent that the memory consumed for Test 10_D_IP and Test 12_D_IP exceeded the total server memory capacity available for the algorithm resulting in the early termination of the code. In both cases, 100% of the original and swap memory were consumed due to the modified IP formulation starting heuristic requiring a view of all the columns and rows of a given optimisation problem before being able to compute an initial feasible solution. The preceding results are, therefore, indicative of the modified IP formulation starting heuristic only being suitable for small-sized optimisation problems. Any optimisation problems exceeding the size of Test 7_IP will result in the code running into memory limitations for a server of the same size as was utilised to solve these test cases. The greedy algorithm is consequently advisable to be employed as the proposed starting heuristic algorithm for the column generation framework. It will allow the end user to solve significantly larger optimisation problems.

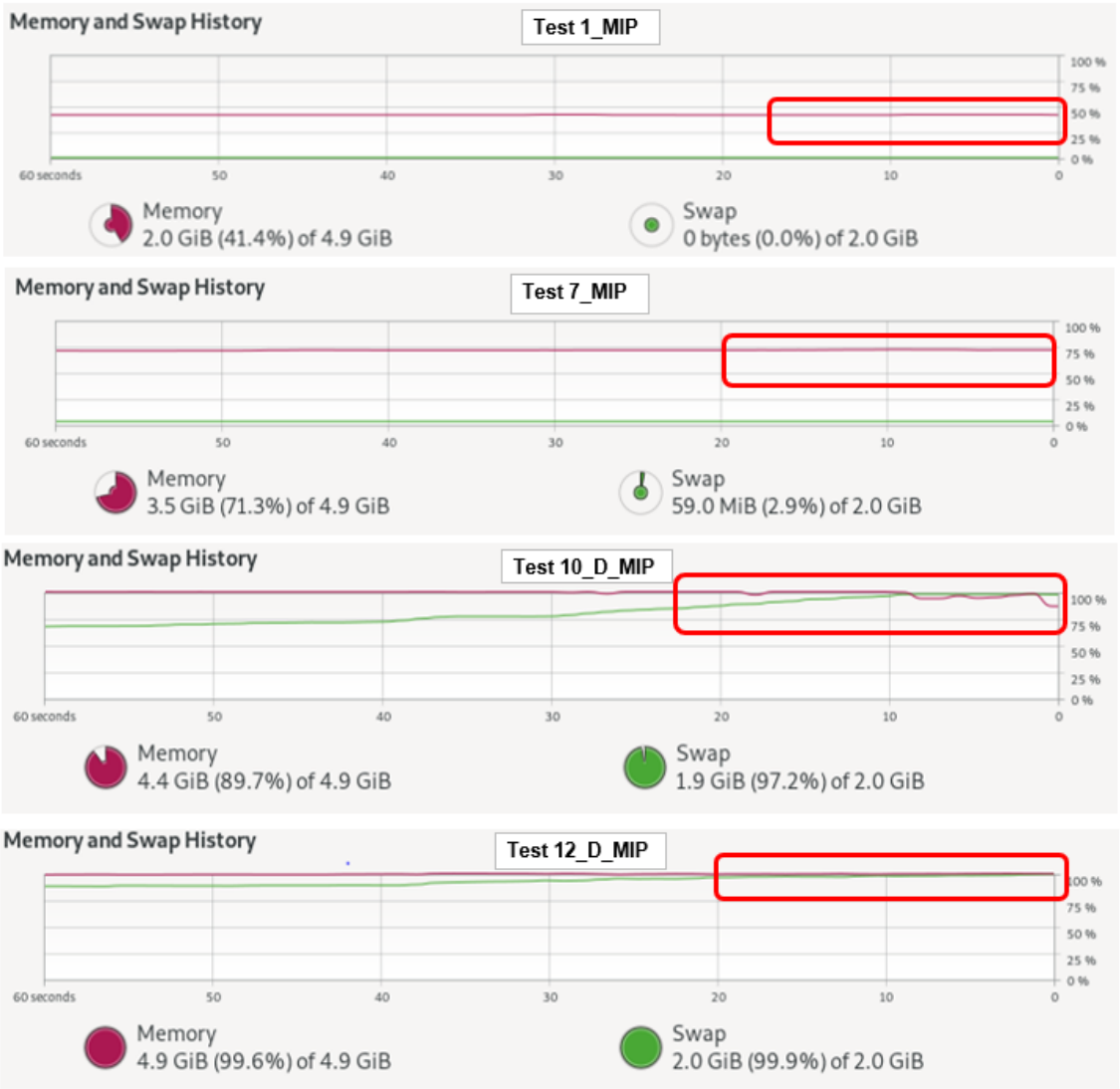

Figure 9.5: Column generation starting algorithm memory consumption tests

Table 9.3 provides a summarised view of the outcomes of the various test instances. As mentioned throughout the above sections, all test instances could be solved using either the greedy algorithm or the modified IP formulation as starting heuristics within the column generation algorithmic framework except for Test 10_D_IP and Test 12_D_IP.

Test Case	Status
Test1	Success
Test1_IP	Success
Test7	Success
Test7_IP	Success
Test10_A	Success
Test10_D_IP	Failed
Test12_A	Success
Test12_D_IP	Failed

Table 9.3: Column generation starting algorithm end state

To emphasize the algorithmic improvements made throughout this thesis, consider Section 9.5. This section explains how the alterations proposed in Chapter 6 improved the solutions generated by the standard column generation algorithm.

9.5 Iterative enhancements to novel column generation approach

An iterative methodology was followed to derive the final column generation algorithm proposed in this thesis. As deliberated throughout Section 6.3 and tested throughout the preceding sections of Chapters 7 and 8, various algorithmic improvements were explored to identify the best possible combination of modifications to the standard column generation methodology, which would produce a superior algorithm capable of solving large-scale product targeting optimisation problems. In order to portray the effect of the iterative algorithmic improvements on the ability of the column generation framework to solve complex product targeting problems, test instances in Table 9.4 and corresponding results seen in Table 9.5 are discussed. As a start to the comparative study, Test 10_D and Test 12_D were executed using the modified IP formulation starting heuristic in combination with the standard column generation algorithm. No algorithmic improvements were made to the standard column generation algorithm for the preceding test instances. In order to improve on the computational outcomes for Test 10_D and Test 12_D, the greedy algorithm starting heuristic was employed in conjunction with the standard column generation algorithm to solve Test 10_C and Test 12_C. The subsequent paragraph will provide more detail regarding the delta improvement obtained from the aforementioned algorithmic updates.

To further enhance the algorithmic performance noted for Test 10_C and Test 12_C, parallel processing, dual smoothing principles, and algorithmic alterations to remove unused columns from the master problem to significantly reduce the computational overhead required by the algorithm were introduced. The effects of the former alterations were investigated by solving Test 10_B and Test 12_B. After critically evaluating the outcomes of Test 10_B and Test 12_B, it was decided to extract the algorithm's capability to remove the unused columns from the master problem as it seemed to have detrimental effects on the ability of the overall column generation algorithm to reach global optimality. The preceding algorithmic combination was utilised to solve Test 10_A and Test 12_A, with this algorithm combination being the one that is proposed to be used when trying to solve large product targeting optimisation problems. Note that for test instance 10, the different algorithmic combinations were applied in solving a product targeting problem consisting of 8000 customers, 25 products, and 3 channels. Test instance 12 was introduced into the comparative study as a slightly scaled version of the product targeting problem containing 10000 customers, 30 products, and 3 channels to be solved. In both instances, the optimisation problems were formulated as loosely constrained.

Test Case	Model Type	Num Cust	Num Prod	Num Chan	Type Constrained
Test10_A	DM	8000	25	3	LOOSE (Kept Cols)
Test10_B	DM	8000	25	3	LOOSE (Removed)
Test10_C	DM	8000	25	3	LOOSE (Kept Cols)
Test10_D	DM	8000	25	3	LOOSE (Kept Cols)
Test12_A	DM	10000	30	3	LOOSE (Kept Cols)
Test12_B	DM	10000	30	3	LOOSE (Removed)
Test12_C	DM	10000	30	3	LOOSE (Kept Cols)
Test12_D	DM	10000	30	3	LOOSE (Kept Cols)

Table 9.4: Column generation input

As eluded previously, Table 9.5 contains the results obtained after solving the above test instances using the various algorithmic combinations. Note that the algorithm was terminated after 12 hours in each test instance to prevent excessive computational time. When focusing on the test outcomes of Test 10_D and Test 12_D, it is apparent that when using the modified IP formulation as starting heuristic to the column generation algorithm, the model cannot compute any solution as it runs into memory limitations. In each of the two mentioned test instances, it was observed that the models already failed at the initialisation step, where the algorithm tried to compute a starting feasible solution to feed into the column generation algorithm. Therefore, the first grouping of algorithms was not a viable combination to further explore if one is focused on solving large-scale product targeting

problems. In an attempt to at least generate a solution for each of the given problem instances, the starting heuristic was updated to use the greedy base algorithm. From the results seen in Table 9.5 for Test 10_C and Test 12_C, it is apparent that by changing the starting heuristic algorithm, the model could generate a solution for each instance. However, it was far from global optimality. For Test 10_C, it is noted that the algorithm could loop through 64 iterations to compute a master objective function value equated to R3326810. It was, however, noted that the algorithm was only able to reach an optimality gap of 92.3% within the allowed period. The memory consumed by the algorithm was 0.37 GB, with 1625 columns and 32183 rows being generated for the given problem instance. Similar results were noted for Test 12_C with the algorithm only completing 55 iterations in the allowed period. An optimality gap of 90.71% was calculated for the given problem instance while generating 40218 rows and 1600 columns. The objective function value obtained for this problem instance equated to R4366740.

After solving the preceding instances, the previously mentioned algorithmic improvements, such as parallel processing, dual smoothing, and removing unused columns, were employed. As it turns out, quite a significant delta lift was noted. When evaluating the results obtained for TEST 10_B, it is clear that due to the improved algorithmic structure, the algorithm could complete a total of 167 iterations in 12 hours. The optimality gap also improved from 92.3% to 28.09% while consuming significantly less memory. The objective function value for the mentioned test instance equated to R3574050, which is remarkably higher than the previous objective function value for the same-sized optimisation problem. When analysing the results for TEST 12_B, the same trend persists, with the algorithm being able to complete 139 iterations instead of 55 in the given period. The optimality gap improved from 90.71% to 11.93%, and the memory consumption dropped from 0.39 GB to 0.24 GB. The algorithmic combination utilised throughout test instance "B" in Table 9.5 seems to be superior to those employed for test instances "D" and "C".

In the last attempt to improve the performance of the algorithm proposed in this thesis, the capability of eliminating unused columns was removed from the computational framework as it was noted that removing columns could detrimentally influence the accuracy of the results being generated. The algorithmic updates were tested by solving Test 10_A and Test 12_A. From the results obtained, it is apparent that the given algorithmic combination outperformed each of the previous combinations. In Test 10_A, the number of iterations was less when compared to Test 10_B as a result of the increased number of columns that were considered (no columns were removed). However, it is noted that the optimality gap reached a value of 12.80% which is drastically better than any of the previous algorithms for the given test instance. The same trend is noted when employing the preceding combination of algorithms to Test 12_A. In this instance, an optimality gap of 11.54% is computed, which is better than the previous attempts while consuming 0.18 GB of memory instead of 0.24 GB. It is, however, essential to note that the algorithm used for test instances "B" does outperform the one applied to "A" as seen in Section 8.2 in some instances. However, given the below outcomes and the results reported throughout Chapter 8, it is apparent that for most problem instances, the greedy algorithm starting heuristic combined with the column generation algorithm enhanced with dual smoothing and parallel processing outperforms the other algorithms quite significantly.

Test Case	Iter	MSTR OBJ (R)	Upper Limit (R)	Lower Limit (R)	INT GAP	OPT GAP (%)	Time (hrs)	Num Cols	Num Rows	Mem Cons (GB)
Test10_A	129	3574660	3821870	3332550	1,07	12,80	12,14	3250	32183	0,17
Test10_B	167	3574050	4636860	3334410	1,07	28,09	12,11	2001	32183	0,24
Test10_C	64	3326810	4320010	3325580	1,00	92,30	12,22	1625	32183	0,366
Test10_D	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed
Test12_A	110	4648410	4938370	4368700	1,06	11,54	12,15	3330	40218	0,18
Test12_B	139	4652750	4961520	4369700	1,06	11,93	12,10	2466	40218	0,24
Test12_C	55	4366740	4697740	4366260	1,00	90,71	12,37	1600	40218	0,39
Test12_D	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed	Failed

Table 9.5: Column generation output

Figure 9.6 provides information regarding the memory response curves for Test 10_C and Test 12_C. Note that the memory response curves for the rest of the problem instances in Table 9.5 have been discussed in the previous sections. When evaluating the response curves for Test 10_C and Test 12_C, it is apparent that in each of the instances, the memory consumed was still well within the operational limits of the server. For example, for Test 10_C, the overall computational framework consumed 65.8% of original memory while only using 0.3% of swap memory. Similar is true for Test 12_C, where 81.7% of original memory was utilised in conjunction with 12.4% of swap memory. When considering the memory response curves for the other test cases mentioned in Table 9.5, it also becomes clear that as the algorithmic modifications are implemented, the memory consumption in each case is significantly reduced. The preceding is precisely what the research study set out to prove, and the proposed column generation algorithm satisfies the set-out objectives.

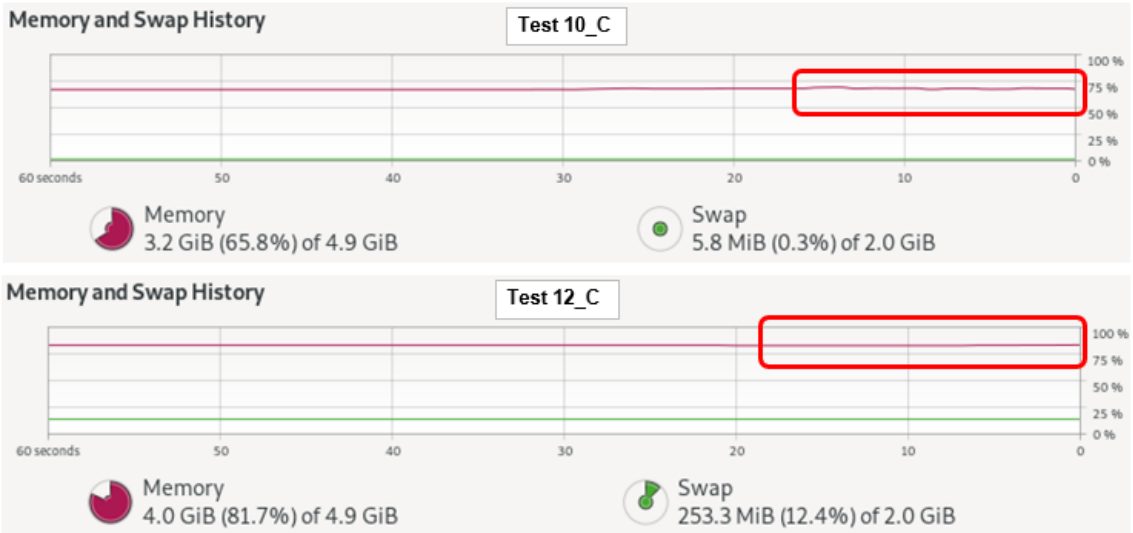

Figure 9.6: Column generation iterative changes memory consumption tests

In Table 9.6, a summarised view is provided to the reader regarding the capability of the various algorithmic combinations in solving the product targeting problem scenarios under consideration. From the below information, it is clear that the grouping of algorithmic combinations encompassed within scenario's "A", "B" and "C" were capable of either computing the global optimal solution or at least generate an answer corresponding to a local optimal region. On the other hand, for the algorithmic combination utilised in scenario D, it is clear that no solutions could be computed.

Test Case	Status
Test10_A	Success
Test10_B	Success
Test10_C	Success
Test10_D	Failed
Test12_A	Success
Test12_B	Success
Test12_C	Success
Test12_D	Failed

Table 9.6: Column generation end state

As part of evaluating the effect of iteratively improving the column generation algorithm, the focus is also set on how the implementation of (6.52) and (6.55) reduces the heading-in and yo-yo phenomenon when applied to Test 10 and Test 12, respectively. In the above section, it is clear that the proposed column generation algorithm significantly reduces memory consumption and computational time required to derive the global optimal solution. Finally, section 9.5.1 discusses some insights regarding why the end user sees a reduction in computational time.

9.5.1 Reducing heading-in and yo-yo effects

Throughout Section 6.3.7, it was identified that the column generation algorithm generally has some limitations when it comes to solving large complex optimisation problems such as heading-in, yo-yo, and tailing-off phenomenons. To address the preceding, algorithmic alterations denoted in (6.52) and (6.55) were introduced to the column generation algorithm to reduce the heading-in and yo-yo phenomenons significantly. In Figure 9.7, the results of a comparative study between Test 10_A and Test 10_C are depicted, where Test 10_A encompassed the algorithmic alterations used to reduce the heading-in and yo-yo effects. At the same time, Test 10_C represented the standard column generation algorithm formulation. The top image in Figure 9.7 depicts the significant improvement in the ability of the algorithm used in Test 10_A to compute the optimality gap compared to Test 10_C. Note that at around 17 iterations, the model was able to exit the heading-in phenomenon allowing it to start reducing the optimality gap from 100% to the desired 0% point. At around 60 iterations, the algorithm used in Test 10_A could already reach an optimiality gap well below 20%. When considering the results for Test 10_C, it is noted that the given algorithm suffered dramatically from the heading-in effect and remained at an optimality gap close to 100% for the period under consideration (12 hours). The bottom image seen in Figure 9.7 provides a clear view of how the proposed column generation algorithm applied to Test 10_A transitioned from the heading-in to the tailing-off effect. Between those two phases, the algorithm generally suffers from the yo-yo effect. Given the implementation of (6.52), the mentioned algorithm exited the heading-in effect quite early in the computational process. After that, (6.55) was responsible for smoothing out the dual values computed by the column generation algorithm to prevent the effect known as the yo-yo effect. Test 10_A does, however, experience some limitations on the tailing-off effect, which was not addressed throughout this thesis and could potentially be investigated throughout future studies.

When considering the outcomes for Test 10_C, it is clear that the standard column generation algorithm entered the heading-in effect and could not exit this phase of the algorithm. The computational process for Test 10_C terminated before the algorithm could even exit the initial phases of the heading-in effect. When critically evaluating the results depicted in Figure 9.7, one could conclude that the algorithmic alterations proposed throughout this thesis do most definitely improve the algorithmic performance of the column generation algorithm and therefore allows the algorithm to solve larger, more complex product targeting problems to optimality within reduced time.

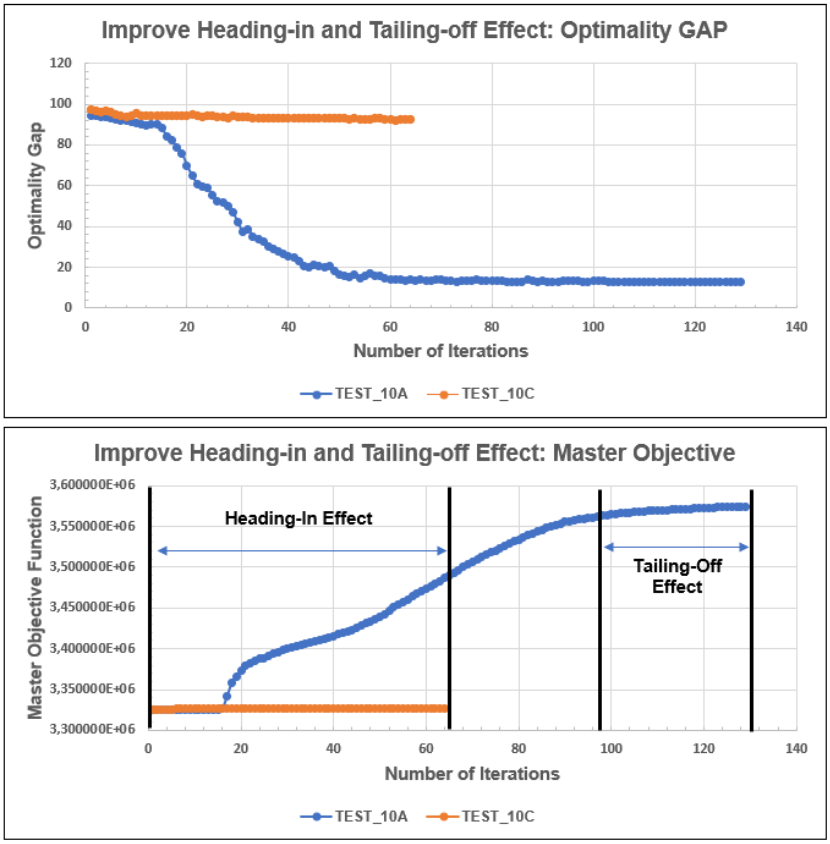

Figure 9.7: Column generation iterative improvements: test 10

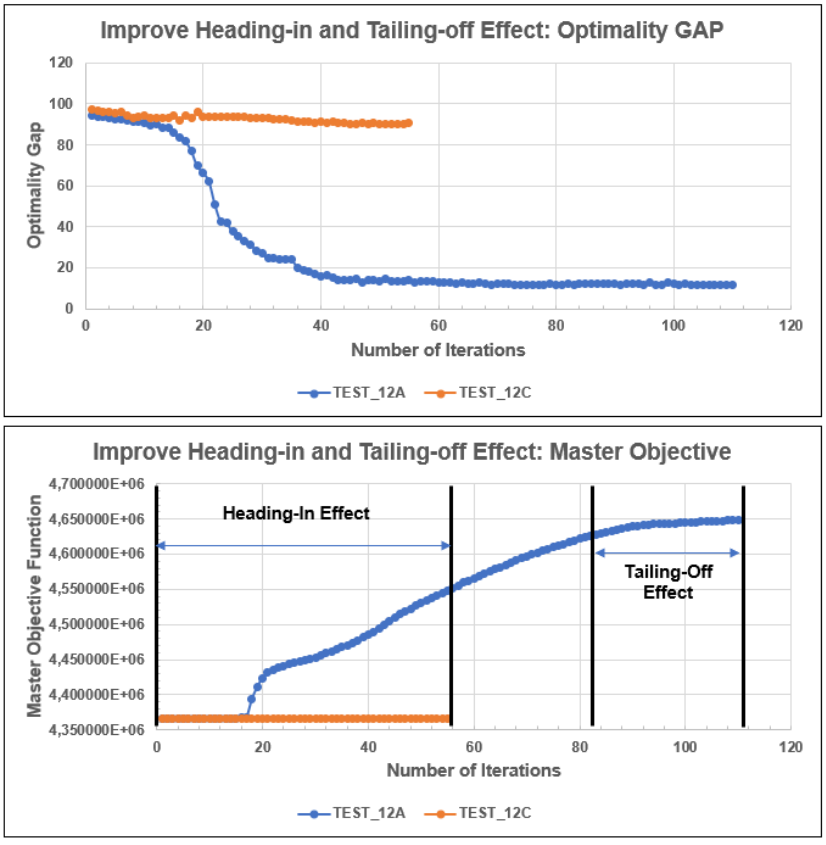

Figure 9.8: Column generation iterative improvements: test 12

Figure 9.8 is provided to substantiate the preceding statements. This figure shows the results generated by solving Test 12_A using the proposed column generation algorithm versus Test 12_C applying only the standard column generation algorithm. By critically evaluating the given results, it is noted that the outcomes of the various test instances are identical to that discussed in Figure 9.7. The top image of Figure 9.8 is indicative of the capability of the modified column generation algorithm to allow the mathematical framework to exit the heading-in effect early on in the calculation process. As a result, Test 12_A was able to start reducing the optimality gap close to the 20 iteration mark. The contrary is true for Test 12_C, where the optimality gap remained close to the 100% range until the point of termination (12-hour computational limit). The bottom part of Figure 9.8 tells the same story, which shows that the modified column generation algorithm can easily overcome the heading-in and yo-yo effects. At the same time, it still suffers quite significantly from the tailing-off effect. On the other hand, the standard column generation algorithm struggles to exit the heading-in effect and, as a result, cannot compute an answer close to global optimality.

From the above insights, it is apparent that the proposed column generation algorithm is capable of reducing the memory consumption of the entire modeling framework and can significantly reduce the computational time by optimising the algorithm's ability to combat the heading-in and yo-yo effects. It is, however, essential to note that the preceding is not the only aspect that improved computational time. By introducing parallel processing into the computational framework, an additional improvement in the compute time is also achieved.

Chapter 10

Summary and conclusion

10.1 Chapter summaries

Product targeting optimisation is a relevant and complex problem for the financial and retail industries. This problem entails the computation of an optimal combination of products or services which should be offered to an identified customer base at the right time of day through the desired communication mediums. In order to construct such a detailed and compendious optimisation problem is an impossible and monotonous task to achieve when employing manual computations. Consequently, detailed mathematical optimisation techniques were utilised to solve this complex optimisation task. When deriving the preceding product targeting mathematical formulation, a multitude of aspects such as customer, product, channel, time of day, cross-sell, and multi-number or email dynamics were considered to ensure that the problem is representative of a realistic real-world product targeting optimisation problem. Increasing the optimisation problem's complexity by accounting for the results of the aforementioned dynamics in the problem is even more challenging to solve.

In Chapter 2, a technical background related to the fundamentals associated with the setup of a product targeting optimisation problem was presented. Aspects such as direct marketing basics, data preparation and generation, machine learning and clustering models development, and decision engine applicability within the campaign environment were discussed. The preceding tried to paint an overall picture of the planning required and the data utilised in a typical product targeting optimisation model. The preceding also provided some insights into the complexities and the effort required to set up and design such an optimisation problem.

Throughout Chapter 3, the focus was placed on the algorithmic advances made by other researchers in this related field of study and the solution algorithms and methodologies applied throughout the literature to solve various formulations of the product targeting problem. Summarizing the contributions made by other research studies provided a clear view of what has already been attempted within the product targeting domain and where areas for improvement could exist. The insights gained from the preceding literature review were utilised as a baseline to steer the development of the proposed solution methodologies as presented in this thesis.

The baseline product targeting IP formulation and the IP formulation was discussed in Chapter 4. The baseline IP formulation provided a summarised view of a specific product targeting problem variation identified in the literature that was used as the departure point for the research study presented throughout this thesis. The initial contribution made in this thesis was to augment the baseline IP formulation with a multitude of additional business, operational, and channel dynamics and constraints to align the problem to a better realistic formulation, as seen in the financial industry. This chapter provided a detailed explanation of each aspect of the IP formulation, emphasising the proposed formulation's complexity.

A detailed review is provided in Chapter 5 on the solution methodologies utilised to solve the

complex product targeting IP formulation proposed in Chapter 4 to global optimality. Insights were provided on the IP formulation's limitations and the branch-and-bound method's inability to solve large problem instances of said formulation. Specific attention was paid to the theory related to algorithms such as the Dantzig-Wolfe decomposition and column generation methods and column generation stabilization techniques available in the literature. Finally, the preceding methods were applied to solve the IP formulation when considering large-scale problem instances. The aim of Chapter 5 was to ensure that the reader comprehends the fundamental principles and reasoning applied in this thesis to arrive at the proposed column generation solution framework.

In Chapter 6, the IP formulation was decomposed using Dantzig-Wolfe decomposition by transforming the problem into a linear programming problem where after the mathematical formulation was split into a master problem (linking constraints) and a multitude of sub-problems (complicating constraints). Additional contributions made throughout this chapter were the introduction of parallel processing and implementation of mathematical principles within the sub-problem objective functions to reduce the heading-in and yo-yo phenomena. The calculation of upper bounds for the column generation algorithm allowed for calculating the optimality and integrality gaps associated with the problem instances. This was necessary to determine how close the proposed solution algorithm got to global optimality.

The results reported in Chapters 7 and 8 provided a platform for analysing the effectiveness and capability of the proposed solution algorithms in solving various test instances of the product targeting optimisation problem. The computational experiments were divided into two testing frameworks: model verification and validation. Model verification was concerned with comparing the test outcomes of the IP formulation (solved using the branch-and-bound algorithm) with that of the column generation algorithm when applied to simplified test instances. Confirming that both algorithms provide the same solution output provided a high level of confidence that the back-end coding of the algorithms is sound and that the results are comparable. Validation testing was focused on evaluating the ability of the various algorithms to solve large-scale problem instances of the product targeting problem. From the preceding test outcomes, the column generation algorithm outperformed the IP formulation when applied to medium to large-sized problem instances. The proposed column generation framework solved problem instances 4 - 5 times larger than the standard branch-and-bound algorithm due to its increased memory utilisation efficiency. However, it was also noted that for small-sized problem instances, the IP formulation would be the desired algorithm to utilise given that the branch-and-bound algorithm could find solutions within a fraction of the computing time compared to the column generation algorithm. The column generation solution framework solved problem instances up to problem sizes consisting of 25000 customers, 35 products, and 3 channels. On the other hand, the branch-and-bound algorithm could only solve problem instances up to a size of 5000 customers, 20 products, and 3 channels.

Throughout Chapter 9, comparative insights were provided regarding the performance of the IP formulation compared to the column generation algorithm for various sizes of product targeting optimisation problems. The focus was set on how the various formulations performed concerning memory utilisation, the ability to reach global optimality, and the number of columns and rows generated when applied to the various test instances. An iterative approach was also followed to introduce the algorithmic enhancements made to the column generation solution framework throughout the thesis and how they improved the overall solution capability of the algorithm. From the results reported in Chapter 9, it is noted that the algorithmic improvements significantly enhanced the ability of the column generation algorithm to solve larger, more complex product targeting problems.

The contributions made throughout this thesis are comprehensive as it provides an excellent baseline for solving significantly larger and more complex product-targeting optimisation problems compared to the standard solution methodologies found in the literature. Although the preceding is true, there

still exists the need for future research to improve on the proposed algorithms to further enhance the capability of the proposed solutions to solve even larger problem instances. Also, additional modeling constraints could be added to the problem formulation to more closely align the model to realistic real-world problem scenarios. A detailed summary of the proposed future work is provided in Section 10.2.

10.2 Future work

To improve the proposed model formulations and algorithms presented throughout this thesis, the following aspects are recommended for inclusion in future work:

1. Future work would be to investigate the feasibility of including channel dynamics such as internet, push notification, and USSD into the modeling framework to move towards a holistic formulation capable of accounting for all aspects of the marketing mix used by financial institutions.

2. Other aspects of investigating would be the development of a heuristic algorithm capable of generating multiple column combinations that could feed into the master problem instead of only being dependent on the sub-problem formulation to generate one column per model iteration. Adding multiple columns simultaneously to the master problem would most definitely decrease the solution time required to find an optimal solution.

3. Possibilities of including segmentation methodologies such as those discussed by Lu & Boutilier (2014) into the complex product targeting framework proposed in this paper would also be worth investigating. Segmenting the customers into various clusters would reduce the number of data points to consider at first glance. However, one would still need to solve each customer's channel and time preference once product assignment has been done on a cluster level. The preceding might consist of multiple smaller optimisation problems to answer different business questions.

4. Potentially designing a heuristic algorithm capable of generating feasible solutions for problem instances where the column generation algorithm fails to provide an answer. The preceding algorithm would be focused on solving significant large-scale product targeting problems outside of the scope of the column generation algorithmic capabilities. For such an algorithm, one would not be too concerned with obtaining global optimality but would instead want to develop an algorithm capable of outperforming random product targeting selection as performed by various other heuristic algorithms.

5. Expanding the column generation algorithm to a branch-and-price framework could reduce the computational time required to arrive at the global optimal solution. However, additional research is required to determine the feasibility of the preceding proposal.

6. When considering customer recency dynamics, the current formulation misses out on potentially very interesting models over longer time horizons that have to plan several contacts simultaneously. The presented model ignores the effect that the current decision has on future time periods and future recency constraints.

7. Potentially adding constraints which governs the amount of products sold via voice for each given time period.

8. Investigating the extent of performance gain that could be achieved in terms of memory consumption and compute time when applying various column generation stabilisation techniques to the column generation algorithm.

9. Investigate the effect of the proposed objective function relative importance variables on solution outcomes. Vary weighting parameters to generate a varied set of solutions that provide real trade-offs between the different parts of the product targeting objective function.

Chapter 11

Appendix A: model parameters and variables

Table 11.1: Basic Model Parameters

Symbol	Definition	Units
p_{ij}	The expected return on investment obtained by the financial institution when offering products j to customers i.	Rand
$c^{(v)}{}_{ij}$	The variable cost incurred by a financial institution when offering products j to potential customers i.	Rand
$c^{(f)}{}_{j}$	The fixed cost incurred by offering project j	Rand
R	Fraction return on investment required by the financial institution	Fraction
B_j	The budget linked to each product j in the campaign process	Rand
$l^{(l)}{}_{j}$	The lower bound for the number of customers that can be assigned to product j	Integer
$l^{(u)}{}_{j}$	The upper bound for the number of customers that can be assigned to product j	Integer

Table 11.2: Novel IP formulation model parameters

Symbol	Definition	Units
p_{iju}	The expected return on investment obtained by the financial institution when offering products j to customers i.	Rand
$c^{(v)}{}_{iju}$	The variable cost incurred by a financial institution when offering products j to potential customers i.	Rand
m_1	Importance weighting term for variable vs fixed cost	Integer
r_{iju}	Probability of reaching a customer on a specific channel	Fraction
$c^{(p)}{}_{iju}$	Enforce preferred medium of contact	Binary
$c^{(s)}{}_{iju}$	Cross-selling weighting variable to force cross-sell to take priority	Fraction
$c^{(q)}{}_{ijuq}$	Probability of RPC for each phone number or email address associated with a given customer i	Fraction
m_2	Channel objective function weighting variable	Integer
$c^{(t)}{}_{iuqt}$	Probability of answer at different time periods of the day for a selected phone number q	Fraction
$c^{(r)}{}_{ijc}$	Financial gain which could be realized when selecting a given cross-sell option	Rand
$r^{(c)}{}_{i}$	Campaign interaction recency indicator	Binary

$P^{(m)}$	Hard limit on the number of products that could form part of the decisioning framework	Integer
$c^{(m)}{}_{\mathrm{u}}$	Channel assignment upper bound for each customer i and product j	Integer
$c^{(l)}{}_{\mathrm{u}}$	Channel assignment lower bound for each customer i and product j	Integer
$l^{(m)}{}_{\mathrm{ju}}$	Upper bound input variable for product j being offered via channel u	Integer
$l^{(n)}{}_{\mathrm{ju}}$	Lower bound input variable for product j being offered via channel u	Integer
E_{ju}	Managing product exclusion for specific channels u and customers i	Binary
$c^{(e)}{}_{\mathrm{iu}}$	Channel preference input parameter	Binary
m_3	Large constant value to ensure that (4.24) holds true when no channel preference has been specified	Integer
$c^{(pr)}{}_{\mathrm{ijuq}}$	Marketing consent tracker for number or email contact mediums	Binary
m_4	Large constant value to ensure that (4.27) holds true	Integer
$c^{(s)}{}_{\mathrm{iju}}$	Indicate if a cross-sell option exists for a certain product j that has been offered to customer i via channel u	Binary
$\mathrm{x}_{\mathrm{wiju}}$	Variable containing extreme point combinations of x_{iju}	Binary
$c^{(d)}{}_{\mathrm{wijuq}}$	Variable containing extreme point combinations of $c^{(d)}_{wijuq}$	Binary
$\mathrm{h}_{\mathrm{wjiuqt}}$	Variable containing extreme point combinations of h_{wjiuqt}	Binary
$\mathrm{o}_{\mathrm{wijc}}$	Variable containing extreme point combinations of o_{wijc}	Binary
$d^{(a)}$	Dual variable linked to constraint set (6.6)	Dual value
$d^{(b)}{}_{\mathrm{i}}$	Dual variable linked to constraint set (6.7)	Dual value
$d^{(c)}{}_{\mathrm{u}}$	Dual variable linked to constraint set (6.8)	Dual value
$d^{(d)}{}_{\mathrm{u}}$	Dual variable linked to constraint set (6.9)	Dual value
$d^{(e)}{}_{\mathrm{ju}}$	Dual variable linked to constraint set (6.10)	Dual value
$d^{(f)}{}_{\mathrm{ju}}$	Dual variable linked to constraint set (6.11)	Dual value
$d^{(g)}{}_{\mathrm{iu}}$	Dual variable linked to constraint set (6.12)	Dual value
$d^{(h)}$	Dual variable linked to constraint set (6.13)	Dual value
$d^{(k)}{}_{\mathrm{j}}$	Dual variable linked to constraint set (6.14)	Dual value

Table 11.3: Model decision variables

Symbol	Definition
x_{ij}	Basic formulation customer commitment decision variable
y_{j}	Product commitment decision variable. Utilised within both basic and IP formulations
$\mathrm{x}_{\mathrm{iju}}$	IP formulation customer commitment decision variable with index u added
$c^{(d)}{}_{\mathrm{ijuq}}$	Best channel selection decision variable
$\mathrm{h}_{\mathrm{jiuqt}}$	Best time of day selection decision variable with specific focus on voice as a channel
$\mathrm{o}_{\mathrm{ijc}}$	Cross-selling commitment decision variable
z_{wj}	Real number weighting decision variable relating to extreme point selection

Table 11.4: Model indexes

Symbol	Definition

I	Index for the number of customers considered with $i \varepsilon I$
J	Index for the number of products considered with $j \varepsilon J$
U	Index for the number of channels considered with $u \varepsilon U$
T	Index for the number of time periods considered with $t \varepsilon T$
Q	Index for the amount of numbers or email addresses linked to individual customers with $Q(i,u) \varepsilon Q$
C	Index for the number of cross-sell opportunities linked to a product j for customer i with $C(i,j) \varepsilon C$
W	Index for the number of columns added to the column generation algorithm with $w \varepsilon W$

Chapter 12

Appendix B: baseline IP formulation summary

12.1 Model objective

$$(B_1) \quad \max \quad \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}(p_{ij}-c_{ij}^{(v)})x_{ij}-\sum_{j\in\mathcal{J}}c_j^{(f)}y_j \tag{12.1}$$

12.1.1 Model constraints

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}p_{ij}x_{ij}\geq R(\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}c_{ij}^{(v)}x_{ij}+\sum_{j\in\mathcal{J}}c_j^{(f)}y_j) \tag{12.2}$$

$$\sum_{i\in\mathcal{I}}c_{ij}^{(v)}x_{ij}\leq B_j, \quad j\in\mathcal{J} \tag{12.3}$$

$$\sum_{j\in\mathcal{J}}x_{ij}\leq 1, \quad i\in\mathcal{I} \tag{12.4}$$

$$\sum_{i\in\mathcal{I}}x_{ij}\geq l_j^{(l)}y_j, \quad j\in\mathcal{J} \tag{12.5}$$

$$\sum_{i\in\mathcal{I}}x_{ij}\leq l_j^{(u)}y_j, \quad j\in\mathcal{J} \tag{12.6}$$

$$y_j, x_{ij}\in\{0,1\}, \quad i\in\mathcal{I}, j\in\mathcal{J} \tag{12.7}$$

Chapter 13

Appendix C: novel IP formulation summary

13.0.1 Model objective

$$(M_1) \quad \max \quad (P + CH + TM + CR) \tag{13.1}$$

$$P = \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} ((p_{iju} - c^{(v)}_{iju}) x_{iju})(r_{iju} + c^{(p)}_{iju} + c^{(s)}_{iju}) m_1 - \sum_{j \in \mathcal{J}} c^{(f)}_j y_j \tag{13.2}$$

$$CH = \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} \sum_{q \in \mathcal{Q}_{iu}} (c^{(q)}_{ijuq} c^{(d)}_{ijuq}) m_2 \tag{13.3}$$

$$TM = \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}} \sum_{u=1} \sum_{q \in \mathcal{Q}_{iu}} \sum_{t \in \mathcal{T}} c^{(t)}_{iuqt} h_{jiuqt} \tag{13.4}$$

$$CR = \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}} \sum_{c \in \mathcal{C}_{ij}} c^{(r)}_{ijc} o_{ijc} \tag{13.5}$$

13.0.2 Model constraints

$$\sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} p_{iju} x_{iju} - R(\sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} c^{(v)}_{iju} x_{iju} + \sum_{j \in \mathcal{J}} c^{(f)}_j y_j) \geq 0 \tag{13.6}$$

$$\sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} c^{(v)}_{ij} x_{iju} \leq B_j, \quad j \in \mathcal{J} \tag{13.7}$$

$$\sum_{j \in \mathcal{J}} \sum_{u \in \mathcal{U}} x_{iju} \leq (1 - r^{(c)}_i), \quad i \in \mathcal{I} \tag{13.8}$$

$$\sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} x_{iju} \geq l^{(l)}_j y_j, \quad j \in \mathcal{J} \tag{13.9}$$

$$\sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} x_{iju} \leq l^{(u)}_j y_j, \quad j \in \mathcal{J} \tag{13.10}$$

$$\sum_{j\in\mathcal{J}} y_j \le P^{(m)} \tag{13.11}$$

$$\sum_{u\in\mathcal{U}} x_{iju} \le 1, \quad j\in\mathcal{J}, i\in\mathcal{I} \tag{13.12}$$

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} x_{iju} \le c_u^{(m)}, \quad u\in\mathcal{U} \tag{13.13}$$

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} x_{iju} \ge c_u^{(l)}, \quad u\in\mathcal{U} \tag{13.14}$$

$$\sum_{i\in\mathcal{I}} x_{iju} \le (1-E_{ju}) l_{ju}^{(m)}, \quad j\in\mathcal{J}, u\in\mathcal{U} \tag{13.15}$$

$$\sum_{i\in\mathcal{I}} x_{iju} \ge (1-E_{ju}) l_{ju}^{(n)} y_j, \quad j\in\mathcal{J}, u\in\mathcal{U} \tag{13.16}$$

$$\sum_{j\in\mathcal{J}} x_{iju} \le m_3(1-c_{iu}^{(e)}), \quad i\in\mathcal{I}, u\in\mathcal{U} \tag{13.17}$$

$$c_{ijuq}^{(d)} \le (1-c_{ijuq}^{(pr)}), \quad j\in\mathcal{J}, i\in\mathcal{I}, u\in\mathcal{U}, q\in\mathcal{Q}_{iu} \tag{13.18}$$

$$\sum_{q\in\mathcal{Q}_{iu}} c_{ijuq}^{(d)} = x_{iju}, \quad j\in\mathcal{J}, i\in\mathcal{I}, u\in\mathcal{U} \tag{13.19}$$

$$\sum_{q\in\mathcal{Q}_{iu}}\sum_{t\in\mathcal{T}} h_{jiuqt} \le m_4 x_{iju}, \quad j\in\mathcal{J}, i\in\mathcal{I}, u=1 \tag{13.20}$$

$$\sum_{t\in\mathcal{T}} h_{jiuqt} \le c_{ijuq}^{(d)}, \quad j\in\mathcal{J}, i\in\mathcal{I}, u=1, q\in\mathcal{Q}_{iu} \tag{13.21}$$

$$\sum_{c\in\mathcal{C}_{ij}} o_{ijc} \le \sum_{u\in\mathcal{U};c_{iju}^{(s)}=1} c_{iju}^{(s)} x_{iju}, \quad j\in\mathcal{J}, i\in\mathcal{I} \tag{13.22}$$

$$y_j, x_{ij}, h_{jiuqt}, c_{ijuq}^{(d)}, o_{ijc} \in \{0,1\}, \quad i\in\mathcal{I}, j\in\mathcal{J}, u\in\mathcal{U}, q\in\mathcal{Q}_{iu}, t\in\mathcal{T} \tag{13.23}$$

Chapter 14

Appendix D: novel column generation approach summary

14.1 Master problem

14.1.1 Model objective

$$(MP_P) \quad \max \quad \sum_{w\in\mathcal{W}}\sum_{j\in\mathcal{J}} z_{wj}(P_{wj}+CH_{wj}+TM_{wj}+CR_{wj}) \quad (=Z) \tag{14.1}$$

$$P_{wj}=\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}((p_{iju}-c_{iju}^{(v)})x_{wiju})(r_{iju}+c_{iju}^{(p)}+c_{iju}^{(s)})m_1-\sum_{j\in\mathcal{J}}c_j^{(f)}y_j, \quad w\in\mathcal{W}, j\in\mathcal{J} \tag{14.2}$$

$$CH_{wj}=\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\sum_{q\in\mathcal{Q}_{iu}}(c_{ijuq}^{(q)}c_{wijuq}^{(d)})m_2, \quad w\in\mathcal{W}, j\in\mathcal{J} \tag{14.3}$$

$$TM_{wj}=\sum_{i\in\mathcal{I}}\sum_{u=1}\sum_{q\in\mathcal{Q}_{iu}}\sum_{t\in\mathcal{T}}c_{iuqt}^{(t)}h_{wjiuqt}, \quad w\in\mathcal{W}, j\in\mathcal{J} \tag{14.4}$$

$$CR_{wj}=\sum_{i\in\mathcal{I}}\sum_{c\in\mathcal{C}_{ij}}c_{ijc}^{(r)}o_{wijc}, \quad w\in\mathcal{W}, j\in\mathcal{J} \tag{14.5}$$

14.1.2 Model constraints

$$\sum_{w\in\mathcal{W}}[\sum_{j\in\mathcal{J}}z_{wj}(\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}p_{iju}x_{wiju}-R(\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}c_{iju}^{(v)}x_{wiju}+c_j^{(f)}y_j))]\geq 0 \tag{14.6}$$

$$\sum_{w\in\mathcal{W}}\sum_{j\in\mathcal{J}}z_{wj}(\sum_{u\in\mathcal{U}}x_{wiju})\leq(1-r_i^{(c)}), \quad i\in\mathcal{I} \tag{14.7}$$

$$\sum_{w\in\mathcal{W}}\sum_{j\in\mathcal{J}}z_{wj}(\sum_{i\in\mathcal{I}}x_{wiju})\geq c_u^{(l)}, \quad u\in\mathcal{U} \tag{14.8}$$

$$\sum_{w\in\mathcal{W}}\sum_{j\in\mathcal{J}}z_{wj}(\sum_{i\in\mathcal{I}}x_{wiju})\leq c_u^{(m)}, \quad u\in\mathcal{U} \tag{14.9}$$

$$\sum_{w\in\mathcal{W}} z_{wj}(\sum_{i\in\mathcal{I}} x_{wiju} - (1-E_{ju})l_{ju}^{(m)}) \leq 0, \quad j\in\mathcal{J}, u\in\mathcal{U} \tag{14.10}$$

$$\sum_{w\in\mathcal{W}} z_{wj}(\sum_{i\in\mathcal{I}} x_{wiju} - (1-E_{ju})l_{ju}^{(n)}y_j) \geq 0, \quad j\in\mathcal{J}, u\in\mathcal{U} \tag{14.11}$$

$$\sum_{w\in\mathcal{W}}\sum_{j\in\mathcal{J}} z_{wj}x_{wiju} \leq m_3(1-c_{iu}^{(e)}), \quad i\in\mathcal{I}, u\in\mathcal{U} \tag{14.12}$$

$$\sum_{w\in\mathcal{W}}\sum_{j\in\mathcal{J}} z_{wj} \leq P^{(m)} \tag{14.13}$$

$$\sum_{w\in\mathcal{W}} z_{wj} \leq 1 \quad j\in\mathcal{J} \tag{14.14}$$

$$0 \leq z_{wj} \leq 1, \quad w\in\mathcal{W}, j\in\mathcal{J} \tag{14.15}$$

14.2 Sub-problem

14.2.1 Model objective

$$(SP) \quad \min \quad (R^C) \tag{14.16}$$

14.2.2 Model constraints

$$\sum_{i\in\mathcal{I}}\sum_{u\in\mathcal{U}} c_{ij}^{(v)}x_{iju} \leq B_j, \quad j\in\mathcal{J} \tag{14.17}$$

$$\sum_{i\in\mathcal{I}}\sum_{u\in\mathcal{U}} x_{iju} \geq l_j^{(l)}y_j, \quad j\in\mathcal{J} \tag{14.18}$$

$$\sum_{i\in\mathcal{I}}\sum_{u\in\mathcal{U}} x_{iju} \leq l_j^{(u)}y_j, \quad j\in\mathcal{J} \tag{14.19}$$

$$c_{ijuq}^{(d)} \leq (1-c_{ijuq}^{(pr)}), \quad j\in\mathcal{J}, i\in\mathcal{I}, u\in\mathcal{U}, q\in\mathcal{Q}_{iu} \tag{14.20}$$

$$\sum_{u\in\mathcal{U}} x_{iju} \leq 1, \quad j\in\mathcal{J}, i\in\mathcal{I} \tag{14.21}$$

$$\sum_{q\in\mathcal{Q}_{iu}} c_{ijuq}^{(d)} = x_{iju}, \quad j\in\mathcal{J}, i\in\mathcal{I}, u\in\mathcal{U} \tag{14.22}$$

$$\sum_{q\in\mathcal{Q}_{iu}}\sum_{t\in\mathcal{T}} h_{jiuqt} \leq m_4x_{iju}, \quad j\in\mathcal{J}, i\in\mathcal{I}, u=1 \tag{14.23}$$

$$\sum_{t\in\mathcal{T}} h_{jiuqt} \leq c_{ijuq}^{(d)}, \quad j\in\mathcal{J}, i\in\mathcal{I}, u=1, q\in\mathcal{Q} \tag{14.24}$$

$$\sum_{c\in\mathcal{C}_{ij}} o_{ijc} \leq \sum_{u\in\mathcal{U}, c_{iju}^{(s)}=1} c_{iju}^{(s)}x_{iju}, \quad j\in\mathcal{J}, i\in\mathcal{I} \tag{14.25}$$

$$x_{iju}, h_{jiuqt}, c_{ijuq}^{(d)}, o_{ijc} \in \{0,1\}, \quad j\in\mathcal{J}, i\in\mathcal{I}, u\in\mathcal{U}, q\in\mathcal{Q}_{iu}, t\in\mathcal{T}, c\in\mathcal{C}_{ij} \tag{14.26}$$

Chapter 15

Bibliography

Agrawal, S. & Agrawal, J. 2015. Survey on anomaly detection using data mining techniques. Procedia Computer Science, 60: 708-713.

Bakar, Z.A., Mohemad, R., Ahmad, A. & Deris, M.M. 2006. A comparative study for outlier detection techniques in data mining. Cybernetics and Intelligent Systems,7 July.

Berkhin, P. 2002. Survey of clustering data mining techniques. http://www.miv.t.u-tokyo.ac.jp /ishizuka/pr-class/clustering_survey(Berkhin2002).pdf. Date of Access: 07 Sept. 2022

Bengio, Y., Lodi, A. & Prouvost, A. 2018. Machine learning for combinatorial optimization: a methodological tour d'horizon. https://casprd.nwu.ac.za/cas/login?service=https%3A%2F%2 Fefundi.nwu.ac.za%2Fsakai-login-tool%2Fcontainer. Date of Access: 7 Sept. 2022.

Bernstel, J.B. 2002. Smart move: creating 'intelligent' database. ABA Bank Marketing, 34: 14-19.

Bhaskar, T., Sundararajan, R. & Krishnan, P.G. 2009. A fuzzy mathematical programming approach for cross-sell optimisation in retail banking. Journal of the Operational Research Society, 60: 717-727.

Bose, I. & Chen, X. 2009. Quantitative models for direct marketing: a review from systems perspective. European Journal of Operational Research, 195: 1-16.

Camilleri, M.A. 2018. Market segmentation, targeting and positioning. University of Malta. (Thesis - PhD).

Cetin, F. & Alabas-Uslu, C. 2017. Heuristic solution to the product targeting problem based on mathematical programming. International journal of production research, 66(1): 3-17.

Coelho, V.N., Oliveira, T.A., COelho, I.M., Coelho, B.N., Fleming, P.J., Guimaraes, F.G., Ramalhinho, H., Souza, M.J.F., Talbi, E. & Lust, T. 2017. Generic pareto local search metaheuristic for optimization of targeted offers in a bi-objective direct marketing campaign. Computers & Operations Research, 78: 568-587.

Cohen, MD. 2004. Exploiting response models - optimizing cross-sell and up-sell opportunities in banking. Information Systems, 29(4): 327-341.

Delanote, S., Leus, R. & Noninon, F.T. 2013. Optimization of the annual planning of targeted offers in direct marketing. Journal of the Operational Research Society, 64(12): 1770-1779.

Desaulniers, G., Desrosiers, J. & Solomon, M.M. 2005. Column generation. New York, NY: Springer.

https://link.springer.com/book/10.1007/b135457. Date of access: 30 Sept. 2020.

Du Plooy, T. 2012. A framework for the planning and integration of out-of-home advertising media in south Africa. University of Pretoria. (Thesis - PhD).

Friberg, D. 2015. An implementation of the Branch-and-Price algorithm applied to opportunistic maintenance planning. Chalmers University of Technology and University of Gothenburg. (Thesis â€“ Masters).

Gamst, M. 2010. Notes on Dantzig-Wolfe decomposition and column generation. https://imada.sdu.dk / jbj/DM209/Notes-DW-CG.pdf. Date of Access. 16 Aug. 2022.

Hellinckx, E. 2004. Customer relationship management: De optimalisatie van de planning van campagnes. Department of Applied Economics, KULeuven. Master's thesis,

IBM Corp. 2015. IBM ILOG Cplex optimization studio Cplex user's manual. https://www.ibm .com/support/knowledgecenter/SSSA5P12.6.2/ilog.odms.studio.help/pdf/usrcplex.pdf. Date of Access: 30 Sept. 2020.

Jain, N. & Srivastava, V. 2013. Data mining techniques. International journal of research in engineering and technology, 2(11): 116-119.

Jassim, M.A. & Abdulwahid, S.N. 2021. Data mining preparation: process, techniques and major issues in data analysis. casprd.nwu.ac.za/cas/login?service=https%3A%2F%2Fefundi.nwu.ac.za%2 Fsakai-login-tool%2Fcontainer. Date of Access: 7 Sept. 2022.

Knott, A., Hayes, A. & Neslin, S.A. 2002. Next-product-to-buy models for cross-selling applications. Journal of Interactive Marketing, 16: 59 - 75.

Koopman, M. 2018. Optimizing the email marketing strategy of an airline using data modelling: a literature study. Vrije University Amsterdam. (Dissertation - Masters).

Lin, L. 2016. Data mining and mathematical models for direct market campaign optimization for fred meyer jewelers. Wright State University. (Thesis - PhD).

Liu, Y., Kiang, M. & Brusco, M. 2012. A unified framework for market segmentation and its applications. Expert Systems with Applications, 39: 10292-10302.

Lubbecke, M.E. 2010. Column generation. Wiley Encyclopedia of Operations Research and Management Science, 1-19.

Lu. T. & Boutilier, C. 2014. Dynamic Segmentation for Large-Scale Marketing Optimization. http://www.cs.toronto.edu/ tl/papers/LuBoutilier_ICML14workshop.pdf. Date of Access: 17 Aug. 2022.

Ma, S. & Fildes, R. 2017. A retail store SKU promotions optimization model for category multi-period profit maximization. European Journal of Operational Research, 260: 680-692.

Mai, L.W.D. 1997. Direct marketing: an analysis of consumers' characteristics and their perceptions of, and attitudes to mail-order specialty food in the UK. Newcastle University, Department of Agricultural Economics and Food Marketing. (Thesis - PhD).

Mitik, M. 2017. Product and channel prediction for direct marketing in banking sector. Middle

East Technical University. (Dissertation - Masters).

Nair, S.K. & Tarasewich, P. 2003. A model and solution method for multi-period sales promotion design. European Journal of Operational Research, 150: 672-687.

Neame, P.J. 2000. Nonsmooth dual methods in integer programming. University of Melbourne, Department of Mathematics and Statistics. (Thesis - PhD)

Ngai, E.W.T., Xiu, L. & Chau, D.C.K. 2009. Application of data mining techniques in customer relationship management: a literature review and classification. Expert systems with applications, 36: 2592-2602.

Nobibon, F.T., Leus, R. & Spieksma, CR. 2011. Models for the optimization of promotion campaigns: exact and heuristic algorithms. https://papers.ssrn.com/sol3/papers.cfm?abstract_id= 1290503. Date of Access: 17 Aug. 2022.

Oliveira, T.A., Coelho, V.N., Souza, M.J.F., Boava, D.L.T., Boava, F., Coelho, I.M. & Coelho, B.N. 2015. A hybrid variable neighborhood search algorithm for targeted offers in direct marketing. Electronic notes in discrete mathematics, 47: 205 - 212

Pal, S. 2006. Column Generation. https://www.cse.iitb.ac.in/ mitra/mtp/seminar/report/colgen.pdf. Date of Access: 16 Aug. 2022.

Pessoa, A.A., Sadykov, R., Uchoa, E., & Vanderbeck, F. 2017. Automation and combination of linear-programming based stabilization techniques in column generation. INFORMS journal on Computing, Institute for Operations Research and the Management Sciences, 30(2): 339-360.

Ralphs, T.K. & Galati, M.V. 2005. Decomposition in integer linear programming. http://www.optimization-online.org/DB_FILE/2004/12/1029.pdf. Date of Access: 30 Sept. 2022.

Raval, K.M. 2012. Data Mining Techniques. International Journal of Advanced Research in Computer Science and Software Engineering, 2(10): 439 - 442.

Reinartz, W., Thomas, J., & Kumar, W. 2005. Balancing acquisition and retention resources to maximize customer profitability. Journal of Marketing, 69(1): 63-79.

Rygielski, C., Wang, JC. & Yen, D.C. 2002. Data mining techniques for customer relationship management. Technology in Society, 24: 483-502.

Savelsbergh, M. 1997. Branch-and-Price algorithm for the generalized assignment problem. Operations Research, 45: 831 - 841.

Smaili, M.Y. & Hachimi, H. 2021. Customer segmentation combining to customer targeting: overview. International journal on optimization and applications, 1(3): 13-16.

Vanderbeck, F. 2005. Implementing mixed integer column generation. Boston, MA: Springer. https://doi.org/10.1007/0-387-25486-2_12. Date of access: 16 Aug. 2022.

Van Niekerk, J. & Terblanche, S.E. 2022. A column generation approach for product targeting optimisation within the banking industry. ORiON, 38(2): 203-229.

Zhao, M. 2005. Mathematical Programming. https://people.orie.cornell.edu/miketodd/or630 /lec20.pdf. Date of Access: 16 Aug. 2022.